FUNDAMENTAL VIBRATIONAL FREQUENCY BY A 3D - FEM PROCEDURE

Howard Lawrence Crego

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

Fundamental Vibrational Frequency
by a
3D - FEM Procedure

by

Howard Lawrence Crego

June 1975

Thesis Advisor: J. E. Brock

Approved for public release; distribution unlimited.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Fundamental Vibrational Frequency by a 3D-FEM Procedure		5. TYPE OF REPORT & PERIOD COVERED Master's Thesis; June 1975 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Howard Lawrence Crego		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93940		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Postgraduate School Monterey, CA 93940		12. REPORT DATE June 1975 13. NUMBER OF PAGES 68
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Naval Postgraduate School Monterey, CA 93940		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Finite Element Method
Vibrations
Oscillations
Frequency

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

This thesis presents a computer program which will solve for the lowest natural vibrational frequency of an arbitrarily shaped three dimensional object. The program was developed using a finite element numerical model which may employ either linear or quadratic isoparametric elements. Necessary instructions and guides for the user are included.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 68 IS OBSOLETE
S/N 0102-014-6601 |
(Page 1)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Fundamental Vibrational Frequency by a 3D - FEM Procedure

by

Howard Lawrence Crego
Lieutenant, United States Navy
B.S.M.E., Purdue University, 1968

Submitted in partial fulfillment of the requirements for the degrees of

MECHANICAL ENGINEER

and

MASTER OF SCIENCE IN MECHANICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
June 1975

ABSTRACT

This thesis presents a computer program which will solve for the lowest natural vibrational frequency of an arbitrarily shaped three dimensional object. The program was developed using a finite element numerical model which may employ either linear or quadratic isoparametric elements. Necessary instructions and guides for the user are included.

TABLE OF CONTENTS

ACKNOWLEDGMENT

The author wishes to gratefully acknowledge the many suggestions and helps he received from the entire FEM group of the Mechanical Engineering faculty at NPS. However, a particular acknowledgment is due Professor J. E. Brock whose patient guidance insured that the development of this thesis was a learning process rather than the mere completion of a degree requirement.

I. INTRODUCTION

This thesis presents and discusses a computer program, called NAT1-3D, which employs the finite element method to set up and solve the elastodynamic equations for the free vibration of a three-dimensional, linearly elastic body so as to emerge with an accurate approximation of the fundamental frequency of free harmonic vibration.

The investigation was originally motivated by the interesting, important, and unsolved problem of the effect of foundation compliance and foundation inertia upon the fundamental natural frequency of a short elastic cantilever beam. It is clear that the usual assumption that the root of such a beam is rigidly fixed does not adequately represent the fact that no foundation is truly rigid. Calculations based upon ideal root fixity thus necessarily result in estimates of the fundamental vibration frequency that are too high. There seems to be no reasonably reliable way of estimating this effect [Ref.1]. [1]

During the course of the work involved in adapting existing computer software to the problem indicated in the last paragraph (in particular adding a dynamic capability which such software did not previously possess) it became more and more apparent that it would be most useful to focus attention upon the development of a general purpose

---

[1] References will usually be cited in this manner; see page 67 for list.

program for the evaluation of the fundamental frequency of three dimensional elastic bodies of any shape. Because of the usual and not unexpected problems of testing and debugging a very complicated digital computer program, time was not available for the study of the root fixity problem. Furthermore, and more importantly, for this problem or any problem in which truly complicated geometry is involved, digital computer programs which rely solely on in-core storage, as does the program developed herein as well as most of its predecessors, are capable of supplying results of marginal accuracy so that parametric studies or studies involving systematic changes in geometry are apt to have limited success. This is because in-core storage limitations do not permit subdividing the body into sufficiently small parts to permit obtaining really accurate results. Accordingly, one of the recommendations made herein is that attention be focussed on modifying this program so as to permit efficient out-of-core storage and manipulation. The present study does demonstrate that the procedural sequence adopted and implemented herein leads to reliable evaluations if the geometry of the vibrating object is sufficiently simple that it may be represented by a reasonably small number of finite elements.

Although the immediately following sections (II, III, IV, V, and VI) discuss theory, sources, and details of the program developed herein, the principal thrust and value of this

thesis is felt to be the detailed instruction provided for the employment of the program NAT1-3D by a new user for the actual evaluation of the fundamental frequency of a three-dimensional elastic body in which he may be interested.

II. THE FINITE ELEMENT METHOD

In recent years a procedure, called the Finite Element Method, has been developed and widely used for a variety of engineering problems involving the analysis of bodies having complicated shapes. In the analysis of the stresses and deformations of an elastic body, using this method (which is generally abbreviated as FEM), the body is subdivided into a (large) number of geometrical subdivisions called finite elements. The adjoining surfaces of these elements are delineated by points called nodes. In a three-dimensional problem each node may have three displacements. Displacements, and, by a process of differentiation, strains at points interior to such a finite element are expressable in terms of nodal displacements by use of so-called shape functions which, roughly speaking, are polynomials in the nodal displacements. The laws of elasticity are used to relate nodal displacements of a finite element to the forces applied at its nodes and the relationship is conveniently expressed in the form of an element stiffness matrix.

An assembly algorithm is used to combine such element stiffness matrices into a so-called global siffness matrix, for the original entire body. This global siffness matrix, hereinafter designated as [BGK], relates nodal displacements for the entire body to nodal forces which may be applied to

it. If there are a large number of finite elements in the body and if each of them has itself a substantial number of nodes, the total number of displacements, three for each node, may be very large.

In elastostatic problems, the nodal forces, i.e., the force components applied to the nodes, are arranged in the form of a vector and the unknown nodal displacement components are similarly arranged, so that the relationship can be expressed in the matrix equation

$$[BGK]\ \{v\} = \{f\}$$

If the size of the vectors {v} and {f} were not great, any of a number of procedures could be used to solve this matrix equation for the unknown vector {v}. However, in practice, the size usually is quite large, and severe problems of storage in an electronic computer have resulted in the development of quite subtle procedures for storing the elements of [BGK] and of {f}.

The most significant features of the matrix [BGK] are that it is positive definite, symmetric, and of banded structure. The last means that there is an upper right and a lower left triangle which consists entirely of zero elements which need not actually be stored if their presence is otherwise suitably noted. These properties permit storing the [BGK] matrix in far less space than if it were to be stored in the form usually used to exhibit square matrices.

Principle features of any FEM computer program are the generation and compact storage of the elements of [BGK], and a so-called "equation solver" which efficiently solves the linear algebraic problem indicated by the equation above. Software to accomplish this has been developed and tested and is readily available. One of the problems facing a person who undertakes to solve an engineering problem by FEM is to select among several competing software programs. TRISOP [Ref. 3] and FIELD [Ref. 4] are the names of programs developed by Professor G. Cantin, LCDR E.A. Leonidas and LCDR G.T. Lew. These programs have been successfully employed at NPS for a variety of investigations. It was decided to adapt and employ appropriate sections of these programs for the purposes of this thesis.

TRISOP and similar programs require a massive input to describe the geometry of the body and the manner in which it is subdivided, to introduce the physical properties of the materials, to represent the manner in which the body is constrained, and to account for the applied loadings. For example, in a typical problem treated in the course of this thesis investigation 250 data cards would be required as input. Since the labor of generating the data for and actually punching these cards is so great and since the likelihood of making at least one crippling error is almost certain, software programs called "mesh generators" have been developed which accept a minimum of input data and

instructions and which produce data cards suitable for subsequent use in an FEM program. Such a mesh generator, called TRIMEG [Ref. 2], developed by LT J.R. Adamek, for use with TRISOP has been used and tested at NPS, and has provided a very useful, almost essential adjunct to the FEM program used for this thesis.

TRISOP does not have a dynamic capability. As will be indicated in the next section hereof, one of the important features of a dynamics FEM program for use in determining natural frequencies is an appropriate mass matrix. Without going into detail, it may be stated that the theory of the FEM provides for the construction of so-called consistent mass matrices for the individual finite elements and their assembly into a global mass mass matrix [BGM]. The computational details are almost exactly the same as for the construction of element stiffness matrices and their assembly into [BGK]. The global mass matrix has essentially the same properties as the global stiffness matrix and requires the same storage.

Much thought was given to the problems of storing such a consistent mass matrix. However, studies by others [Ref. 7] have indicated that when evaluating the fundamental harmonic frequency of free vibrations of an elastic object, use of a lumped-mass form of the mass matrix gives results as good as or better than those obtainable by using a consistent mass matrix. The reasons for this are not clear,

but the advantages are. A lumped-mass mass matrix is diagonal in form and can be stored as a vector; briefly, using the lumped-mass form results in a dynamics problem requiring very little more storage than an elastostatic problem requires.

Accordingly it was decided to employ the lumped-mass form and the question arose of how optimally to obtain such a mass matrix. The procedure which finally evolved was to construct the element consistent mass matrix and to diagonalize it before assembly into the global mass matrix.

The word diagnalize in this context is not intended to mean the mathematical process of diagonalization by means of solving the complete eigenvalue problem for the matrix M. Instead, it merely means replacing the consistent (element) mass matrix by a diagonal matrix corresponding to an appropriate division of the mass of the element into submasses concentrated at the nodal points.

The assembly was done in the core storage space reserved for assembling (and later operating upon) the global stiffness matrix, and the assembly algorithm is exactly the same for both matrices. The assembled mass matrix is diagonal and its nontrivial elements are quickly stored elsewhere in the form of a vector.

The details of diagonalizing the element mass matrix deserve a brief description, explanation, and plausible justification. Aside from the fact that it is obviously

desirable to employ a procedure which involves a simple algorithm and results in the same diagonalized element mass matrix regardless of renumbering of nodes, a reasonable and plausible distribution of the total element mass into mass points at the nodes is also required. This matter is discussed in somewhat greater detail on page 29.

Accordingly appropriate segments of the programs TRISOP and FIELD were selected and modified so as to incorporate the construction of a lumped-mass mass matrix as outlined in the preceding paragraphs and so as also to incorporate the operations of LDLT decomposition, matrix (Stodola) iteration, convergence testing, and Rayleigh quotient evaluation discussed in the following section hereof. The resulting program is called NAT1-3D.

III. METHOD OF SOLUTION

With the use of the [K] and [M] matrices provided by the finite element formation, as discussed previously, the free vibration problem can be formulated as

$$[M] \{\ddot{v}\} + [K] \{v\} = \{o\} \qquad (1)$$

The remarks which follow immediately represent standard procedure in vibration theory and are intended merely to recall to the reader the operational sequence which leads to the desired result. We are concerned with the lowest frequency harmonic vibrations which the system may execute. By assuming that the vector $\{v\}$ of time-varying displacements has the form

$$\{v\} = \{v^*\}\cos(wt)$$

(where $\{v^*\}$ is a vector of constant quantities representing the amplitudes of the motions of the nodal points) and substituting this equation into equation (1) above, the cosine term may be cancelled out, and, upon dispensing with the asterisk and henceforth regarding $\{v\}$ as a vector of constants, we arrive at the form

$$[K] \{v\} = \omega^2 [M] \{v\}$$

It may be shown that if an arbitrary vector $\{v_i\}$ is introduced on the right side and this equation is solved for a new vector, i.e., if we solve for $\{v_{i+1}\}$ in the equation

$$[K]\ \{v_{i+1}\} = \omega^2[M]\ \{v_i\}$$

then $\{v_{i+1}\}$ is a better approximation to the theoretically correct vector for the lowest frequency than was $\{v_i\}$. Furthermore, if the vectors are suitably and similarly normalized, the scalar constant of normalization approaches more and more closely to the correct value of the square of the lowest frequency. This process of matrix iteration is frequently called Stodola-iteration after the Swiss engineer who introduced the method.

Although the scalar constant of normalization does provide an increasingly good evaluation of the desired value of ω_1^2, at any stage of the iteration a more accurate estimate may be obtained from the Rayleigh quotient

$$\omega_1^2 \tilde{=} Q = \{v\}^T\ [k]\ \{v\}\ /\ \{v\}^T\ [M]\ \{v\}.$$

Performing this operation after a reasonable number of iterations provides the best currently available value for ω_1^2 and, presuming that the value so obtained is "close" to the values previously obtained during the Stodola iteration, provides a check on the accuracy and the validity of the entire computational process.

With systems having only a few degrees of freedom it is customary to solve for $\{v_{i+1}\}$ by obtaining the inverse of the matrix [K]. However, the desirable property of the [K] matrix as obtained by the finite element method, namely its banded character which permits compact storage, is not possessed by the inverse $[K]^{-1}$; furthermore the operation of obtaining an inverse is computationally extravagant. Accordingly, it is very advantageous to represent [K] in the so-called LDLT form, i.e., to decompose [K] into the matrix product

$$[K] = [L]\,[D]\,[L]^T$$

where [L] is a lower triangular matrix having unit diagonal elements (which need not be stored) and [D] is a diagonal matrix. This decomposition is a standard operation which is not difficult to perform and the matrices [L] and [D] may be stored in the space occupied by [K].

The normalization which is used is the so-called Euclidean normalization for which the unnormalized vector {v} is divided throughout by the square root of the sum of the square of its elements.

The iteration proceeds as follows. Having the normalized vector $\{v_i\}$ (either as a result of having made a previous iteration or, the first time through, as the result of a guess or enlightened estimate) the vector

$$\{g\} = [M]\,[v_i]$$

is obtained simply by matrix multiplication. Then the vector {h} is determined from the equation

$$[K]\{h\} = \{g\}$$

by a forward and backward substitution process making use of the desirable properties of the LDLT decomposition. Then {h} is normalized so as to produce the next iterant $\{v_{i+1}\}$, the reciprocal of the constant of normalization being an approximation to ω_1^2. This process is repeated until convergence is satisfactory. In the computer implementation of this procedure, described later, convergence was assumed to be satisfactory when two successive such evaluations of ω_1^2 differed by less than 10^{-6}. At this point, the Rayleigh quotient was evaluated in the form

$$Q = ([L]^T \{v\})^T [D] [L]^T \{v\} / \{v\}^T [M]\{v\}$$

Although the procedure described above could be employed with nondiagonal [M], the fact that it was decided to diagonalize [M] in order to minimize core storage requirements also made the computations involving [M] very simple and economical.

IV. PROGRAM ARRANGEMENT

To facilitate the adaptation of existing mesh generating schemes and to incorporate the strong points of existing software the general arrangement of the program is modular in nature. This also allows future changes to be accomplished more easily if desired.

The actual solution of a problem requires two basic steps: mesh generation and frequency evaluation.

A. MESH GENERATION

Mesh generation is the process in which the object to be analyzed is divided into an assemblage of small elements. It is a laborious and tedious task that should not be rushed into. Intuition guided by an intelligent study of the geometry of the object is often the best approach to use in choosing a mesh.

TRIMEG (Ref. 1, which is discussed in greater detail in Appendix C) is a computer program that performs mesh generation for three dimensional objects. The program automatically discretizes a given geometry and upon command will punch data decks containing nodal coordinate and element connectivity information. The solution routine presented in this thesis was written to accept data in the format of TRIMEG's output. The use of TRIMEG is not a necessity and any source may be used for the input data. However, the use of

TRIMEG to generate data cards for use with NAT1-3D is strongly recommended to save effort and to reduce the likelihood of error. The reader is advised to study and understand TRIMEG; Ref. 1.

B. FREQUENCY EVALUATION

1. Common Statements

Throughout the development of this routine a prime objective was to make it as simple as possible to use. Simplicity can however limit flexibility. To enable the user to avoid requesting excessive core space by adjusting the core requirements to meet the needs of his particular problem, the large data blocks are placed in easily changed COMMON storage areas. This adjustment allows the use of more nodes for problems with smaller band widths.

There are four labeled COMMON statements which may be used to modify core storage requested by NAT1-3D; this is done to avoid requesting more core than necessary and thus results in obtaining optimum job priority. The locations of the cards and size of the arrays are indicated in the listing in Appendix B. The following is a tabulation of the statement names and the subroutines in which they appear.

BIGS	MAIN, MERGE, QUAD, FORM, FASTEN, LTLD, SOLV
ENTER	INPUT, MERGE
ITER	MAIN, NORM
SOL	MAIN, SOLV

In addition to modifying the COMMON statements the dimension of the vector FMODE which appears in the main routine should be calculated and introduced in the appropriate DIMENSION statement.

2. Input

The required data consists of six basic card groupings. Their specific formats are listed in Appendix B.

a. TITLE (one card)

Any desired problem title may be used.

b. PROBLEM DATA (one card)

The card contains the basic problem parameters. Here the user chooses the type of shape functions that are desired and indicates the number of Gauss points he wishes to use in their integration.

c. MATERIAL PROPERTIES (one card per material)

These cards contain the elastic property data for the isotropic materials used in the object. The number used to represent a material is arbitrary but it must be the same numbering scheme used in the mesh generation.

d. NODE CARDS (one card per node)

These cards, which may be punched by TRIMEG, contain the node numbers and their global coordinates.

e. CONNECTIVITY MATRIX (two cards per element)

These cards, which may be punched by TRIMEG, contain the detailed arrangement of all the contributions of each element to the nodal displacements. As the name implies,

this matrix connects the elements and serves as the controlling agent which distributes the contribution of each element to the global stiffness and mass matrices.

f. BOUNDARY CONDITION CARDS (one card per boundary node)

The user may impose boundary conditions by restraining the motion of any node in any coordinate direction. The boundary condition cards indicate the node and the coordinate direction to be fixed.

V. EXAMPLE PROBLEMS

The method usually used to test computer programs is to solve a problem that has a known exact solution to serve as a comparison gauge for the computer solution. This presents a difficulty as the author is unaware of a truly three dimensional classic solution for natural frequency determination. Problems A and B, below may seem overly simple but they do have known approximate solutions and serve to establish a measure of confidence in the integrity of the routine.

The approximate solutions are based on Euler beam theory or on the more elaborate Timoshenko beam theory, both of which involve replacing the equations of three dimensional elasticity (theoretically required to describe the problem) by greatly simplified equations.

A. CANTILEVER BEAM

The cantilever beam is a basic design structure which may vary in form from the smallest man-made gear tooth to the largest trees produced by mother nature. The difficulty of analysis also may vary as drastically. The inclusion of effects such as shear and rotary inertia significantly complicates the frequency equation.

The specific structure used in this problem was a beam six inches long having a uniform one inch square cross section. Subdivision is indicated in Figure 1. The results were as follows:

Euler Beam Theory	ω = 5700 rad/sec
NATl-3D Results	ω = 5664 rad/sec
Ratio NATl-3D/Euler	0.994

This agrees with Timoshenko theory which predicts a frequency correction for shear and rotary inertia of nearly unity for this geometry [Ref. 8]. Figure 1 shows the convergence of the solution as the number of nodes is increased, i.e., the effect of finer subdivision. Note however, that the finer discretization involves only the "Z" direction. It would be of interest also to investigate finer subdivisions in the "X" and/or "Y" direction. However, this would involve substantial increases in bandwidth.

B. SIMPLY SUPPORTED BEAM

The effects of shear deflection and rotary inertia are more pronounced in a simply supported beam having the same geometry as the previous cantilever.

Euler Beam Theory	ω = 16,000 rad/sec
NATl-3D	ω = 14,250 rad/sec
Timoshenko Beam Theory	ω = 14,000 rad/sec

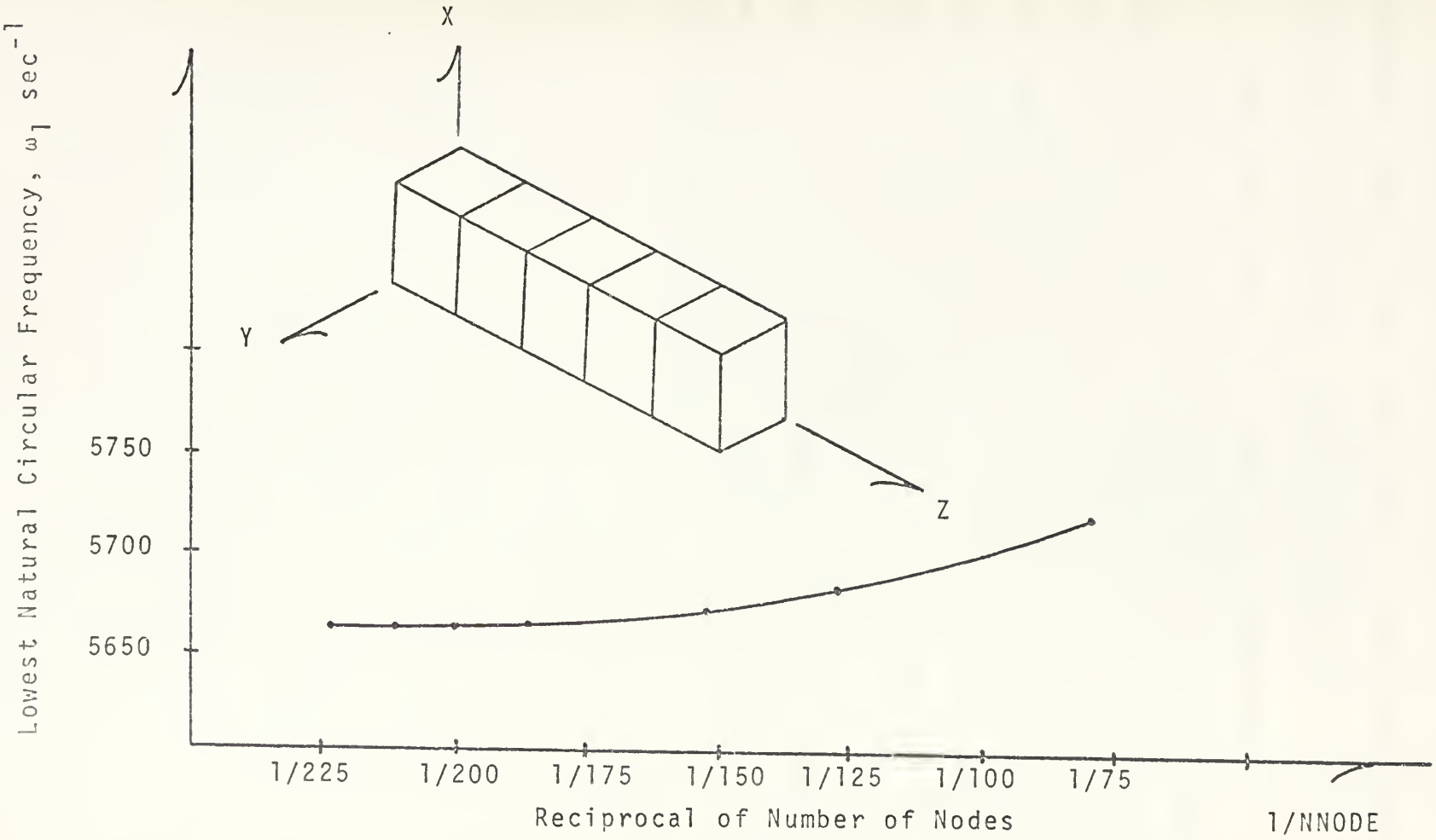


Figure 1. Convergence of Solution as Number of Nodes Increased.

The Timoshenko solution is based on an assumed shear constant value of 0.6. This value is the approximate middle of the accepted range of values [Ref. 8].

C. CANTILEVER BEAM WITH COMPLIANT FOUNDATION

Figure 2 represents a mesh which could be used to investigate the effects of root fixity on the natural frequency of a cantilever beam having imperfect root fixity. The boundary conditions would be applied to the outer wall surfaces thus allowing the root to deflect naturally. The material properties of the wall and the beam can also be varied to measure the effects of dissimilar materials. A final solution was not obtainable due to the large core requirements arising from even the rather coarse discretization indicated in Figure 2.

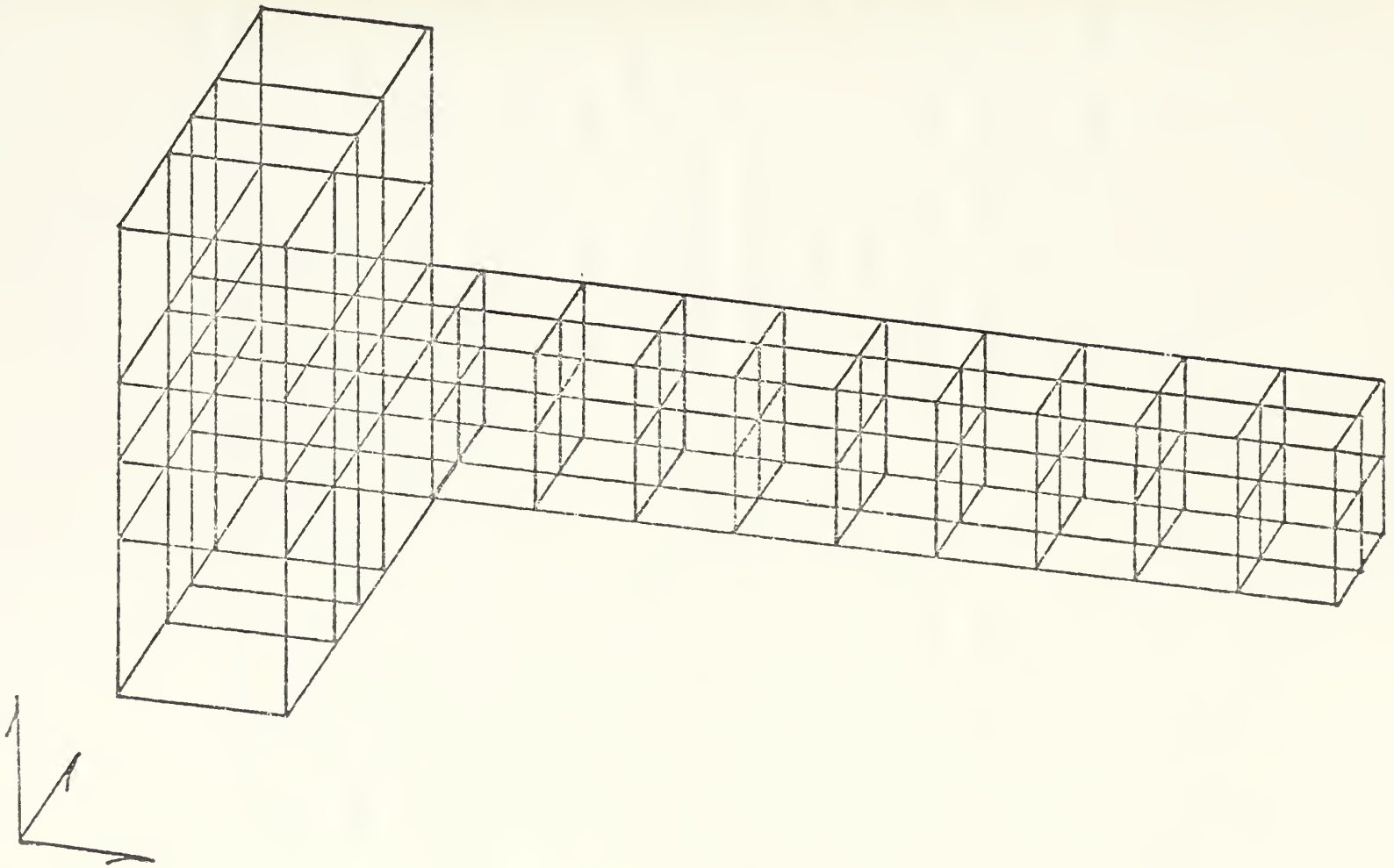


Figure 2. Discretization of Cantilever to Investigate Root Fixity Effects.

VI. DISCUSSION

A. MASS MATRIX

As noted in chapter II a lumped diagonal mass matrix was chosen for use in the solution routine. The choice was based upon the reduced core requirements and computational advantages inherent to the diagonal matrix. A decision was made to form the lumped matrix from the consistent matrix because the numerical evaluation of the consistent mass matrix parallels the evaluation of the stiffness matrix allowing the two formulations to be coded simultaneously. Lumping is accomplished by replacing the diagonal entry in each row of the consistent matrix with the sum of the entries in that row and zeroing the non-diagonal elements. The result could be termed a consistent, lumped matrix.

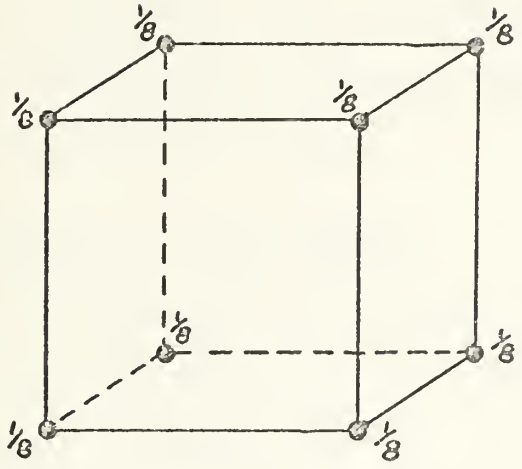
Reference 7, previously cited, indicate that this form of mass matrix yields excellent results for the lowest frequency. It is of some interest at this point to examine the point-mass distribution which results from this process in the case of an element in the form of a rectangular parallelepiped, a case for which physical intuition provides some indication of whether or not the substitution is reasonable.

Figure 3 shows the resultant mass distribution for both linear and quadratic elements. The distribution for the linear element appears logical. The quadratic element however is not at all obvious. The allocation of negative mass at the element corners would seem to conflict with common sense. These results are however, not without precedent. Negative corner values result when imposing uniform body forces on rectangular elements in a pattern that is nearly identical to that shown in Figure 1 of Ref. 6 (p. 168).

B. SHAPE FUNCTIONS USED

In its earlier versions NATI-3D accommodated quadratic shape functions only. This was a compromise between the limitations of linear functions in representing complex geometry and the large core size requirements of cubic elements. A later inclusion of linear shape functions resulted in the observation that they were "too stiff." The answers they gave for the fundamental frequency were high when the same number of nodes and same nodal arrangement was used that corresponded to an accurate solution based on quadratic functions.

This is to be expected since, for the same subdivision into elements, the shape functions for quadratic elements permit a closer representation of the actual deformation pattern than is permitted by using linear elements. In effect this corresponds to both types of elements implicitly



Distribution of Mass, Rectangular Parallelepiped, Linear Functions

Distribution of Mass, Rectangular Paralleleniped, Quadratic Functions

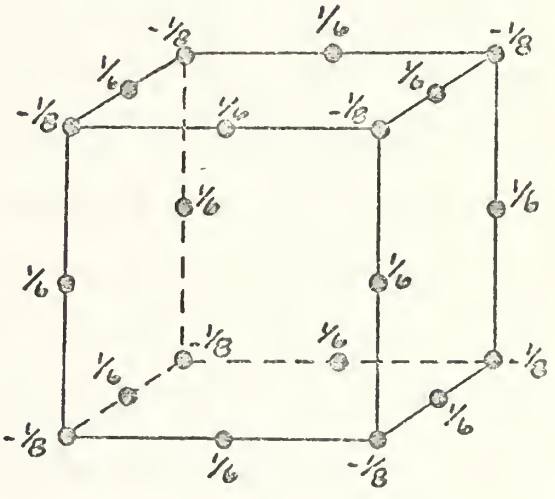


Figure 3. Allocation of Lumped-Mass Mass Matrix.

introducing fictitious and undesirable restraints, which increase the calculated frequency above its correct value; however, these restraints are greater and have a more serious effect in the case of linear elements than they do in the case of quadratic elements.

C. NUMBER OF GAUSS POINTS

If N represents the number of Gauss points, the Gaussian technique used in this routine will exactly integrate a polynomial of degree (2N-1). It has been noted however, in calculations employing NAT1-3D that the minimum number of Gauss points which can successfully be used with quadratic elements is three. When fewer than three Gauss points are used it is found that accuracy of the frequency evaluation is seriously decreased. It is conjectured that this behavior is related to the method of introducing the boundary conditions (by multiplying selected diagonal elements of [BGK] by a large factor, "WELD"). The calculations which showed the undesirable behavior had WELD = 10^{10}. It would be interesting to look into this matter more fully by reducing the value of WELD, by considering alternate methods of introducing boundary fixity, and in other ways. Briefly, we do not presently have a satisfactory understanding of the effect of modifying the number of Gauss points except to observe that using NGP = 2 gave unsatisfactory results, while using NGP > 3 gave results which were felt to be no more accurate or reliable than those obtained by using NGP = 3.

D. CONVERGENCE

Our experience has shown that the Stodola iteration converges with amazing rapidity, three to four iterations usually being sufficient. This can be attributed to the fact that the structures considered had lower frequencies which were widely separated from the second frequency. It is expected that objects whose lower frequencies are not separated will require more iterations to converge.

It should be noted that the "initial guess," being a unit vector, contains elements of both transverse and axial vibrations. If the lowest mode were an axial mode it would be reflected in the corresponding elements of the stiffness matrix and the iteration should converge to the actual lowest frequency be it an axial or transverse mode. Correct solutions have been obtained for transverse vibration using a pure axial mode as an initial guess.

E. PROGRAM LIMITATIONS

It could be considered overkill to use this routine on a truly two dimensional plane strain or plane stress problem. During the development of this routine a 2-D version was written but it required exactly the same work on the user's part in the form of data preparation and further required the user to recognize that his object could be modeled as a plane stress/strain case. Upon considering that an original

goal was to limit the assumptions and idealizations that the user has to make, only the three dimensional routine is being presented.

In its present format the program is limited by core size. This problem arises when dealing with complex geometries which require several layers of elements in more than one direction. Multidirectional layering results in a dramatically rapid growth in the band width of the stiffness matrix which is the major core consuming element of the entire routine.

VII. RECOMMENDATIONS

A. ELIMINATE STORAGE LIMITATIONS

The first and most obvious recommendation for further improvement of the routine is to remove the core limitations imposed on the working area. This can be accomplished by the use of direct access devices and a block solving routine which sectionalizes the matrices and works in core with only one section at a time.

It should be noted that the implementation of these modifications will be a difficult task. Substantial coding changes will be required and the difficulties encountered are anticipated to be orders of magnitude greater than those associated with the in-core routine.

B. PARAMETER STUDY

It is normally a part of verifying and testing a FEM program to perform parametric studies on several problems having known exact or approximate results. Time did not permit doing such an extensive investigation with NAT1-3D and it is recommended that appropriate attention be given to a systematic study of the effects of varying the values of NGP, WELD, and the degree of the shape functions and of varying the fineness of the ultimate discretization in each of the three coordinate directions. The literature on FEM describes many such studies and can be used to guide such an

investigation. However, it may be stated that such previous investigations have not led to any significant understanding of the effects of varying parameters in general. Apparently, as of yet, it is important to carry out such an investigation for each new FEM code.

APPENDIX A

PROGRAM DESCRIPTION

A complete disection and detailed description of the program is not practical. This appendix is intended to contain information which will assist both a reader who desires to pursue a more in-depth study of the problem and a reader who wishes only to make an intelligent casual use of the program.

A. TERMINOLOGY

The following is a listing and brief description of important constant, vector, and matrix names. Functions, definitions and dimensions are given if applicable.

1. Constants

FACT	Value of the Euclidean norm of the mode vector, used to determine iteration convergence
NBAND	Half band width of the stiffness matrix
NEL	Number of elements
NEQ	Number of equations, NEQ = 3 x NNODE
NGP	Number of Gauss points used in the Gaussian quadrature formula
NMAT	Number of materials
NNBC	Number of bounded nodes
NNODE	Total number of nodes
NPEL	Number of nodes per element

2. Vectors (size)

B, EMODE, FMODE (NEQ)	Working vectors used to store the mode shape during iteration and while evaluating the Rayleigh quotient
BGM	Diagonalized mass matrix stored as a vector

3. Matrices (dimensions)

BGK(NEQ,NBAND)	Global or system stiffness matix
COARD(NNODE,3)	Matrix containing global coordinates of nodes
ENW(3*NPEL,3*NPEL)	Element stiffness or mass matrix
NCONN(NEL,NPEL+1)	Connectivity matrix
NBC(30,4)	Matrix of bounded nodes (i.e., nodes on which boundary conditions are to be imposed) and their boundary conditions

B. PROGRAM ARRANGEMENT

As indicated in Figure 4 the routine is of modular construction. This format allows for maximum ultilization of existing software and facilitates future revisions.

1. MAIN

The first segments of the MAIN program initiate the calculation of the large system matrices. Data is passed to the various subroutines via common statements. The iterative portion of MAIN is based upon the decomposition of BGK and the repeated solution of the matrix equation $[BGK]\{v\} = \omega^2[BGM]\{v\}$. Once convergence has been attained MAIN controls the calculation of the Rayleigh quotient which serves as both a check and a refinement of the frequency obtained by the iteration.

ARRANGEMENT OF NAT1-3D

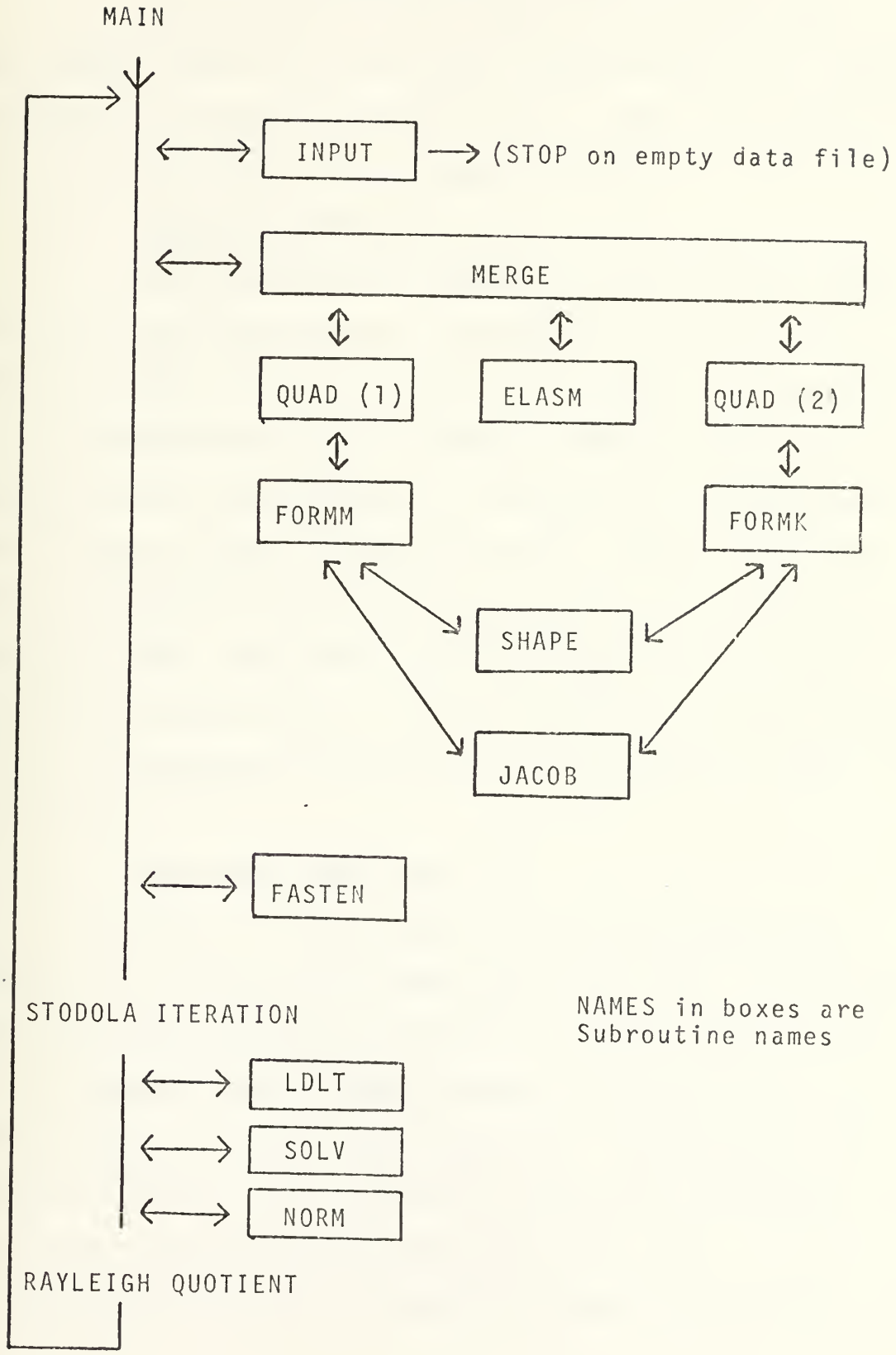


Figure 4

2. Subroutine INPUT

INPUT reads and echo checks the problem data. The most common source of errors in a large routine is a simple typographical error in the data deck. INPUT also computes the half band width of the system matrices.

3. Subroutine MERGE [Ref. 4]

MERGE combines the contributions of each element and distributes them to the corresponding global matrix, [BGK] or [BGM]. The distribution allocation is controlled by the connectivity matrix (NCONN). MERGE serves in a sequence controlling capacity throughout the calculation of the system stiffness and mass matrices. BGM is computed first and during its assembly is stored in the region set aside for BGK. Calculation of the weight of the object provides a quick check on the mass matrix by giving the user an evaluation which he may compare to the actual weight of the object.

4. Subroutine QUAD [Ref. 4]

QUAD is divided into two basic sections which differ in the form of the shape functions used. Computing the mass matrix requires evaluation of the shape functions while computing the stiffness matrix requires evaluation of the derivatives of the shape functions. QUAD selects the weighting factors and integration points that correspond to the number of Gauss points desired by the user. It then calls the appropriate entry of Subroutine FORM which evaluates the

integrand of the Gaussian quadrature formula. Next QUAD multiplies the result by the Gaussian weighting factor and computes the summation. Diagonalization of the mass matrix is also accomplished in QUAD.

5. Subroutine ELASM [Ref. 3]

ELASM forms the constitutive elasticity matrix for each element based on its material which is assumed to be linear, isotropic and elastic.

6. Subroutine FORM [Ref. 4]

FORM evaluates the product of the shape functions or the product of their partial derivatives at each selected Gauss point in the local coordinate system. Dual entry points are used to distinguish between the computations necessary to form the elemental mass matrix and the elemental stiffness matrix.

7. Subroutine SHAPE [Ref. 4]

SHAPE performs the actual evaluation of the shape functions and their derivatives with respect to the local coordinate system. Provisions are included for both linear and quadratic shape functions.

8. Subroutine JACOB [Ref. 4]

JACOB evaluates the Jacobian matrix as well as its inverse and its determinant.

9. Subroutine FASTEN

FASTEN spatially fixes the bounded nodes in the desired coordinate direction by multiplying the appropriate diagonal elements of [BGK] by 10^{10} so as effectively to suppress the associated deflection component. This is a commonly used method, in FEM analysis, of introducing fixity at boundary nodes.

10. Subroutine LDLT [Ref. 5]

LDLT decomposes the system stiffness matrix [BGK] into the products of three matrices: a lower triangular matrix [L] having unit values on the diagonal, a diagonal matrix [D], and $[L]^T$ the transpose of [L]; viz. $[BGK] = [L][D][L]^T$. In the process [BGK] is destroyed and replaced by the elements of [D] and the nontrivial elements of [L]. The diagonal matrix is stored in the first column and the unit triangular matrix is stored in the remaining space.

11. Subroutine SOLV [Ref. 5]

SOLV performs a forward substitution followed by a back substitution to solve the system [BGK] {X} = {B}. The answers, {X}, are returned in vector {B}.

12. Subroutine NORM

NORM calculates the Euclidean norm of the mode vector. This norm is equal to the square root of the sum

of the squares of the element values in the vector. Each element in the vector is then divided by the norm which results in normalizing the vector.

C. CORE ESTIMATION

The mesh generating scheme chosen by the user should inform him of the number of nodes and the bandwidth that have resulted from the discretization. TRIMEG clearly lists this data. With the aid of the following equation these figures may be used to estimate the core requirements which result from the mesh.

$$\text{"K" required} = 110 + \frac{24 \times \text{NNODES} \times (\text{NBAND} + 4)}{1024}$$

The resultant answer includes the necessary buffers and can be used directly as the region specified on the FORTCLG control card.

```
//  STANDARD GREEN JOB CARD
//  STANDARD EXEC FORTCLG CARD
// STANDARD FORT.SYSIN DD CARD
C*****************************************************************SKIP0010
C*****************************************************************SKIP0020
C                                                                 SKIP0030
C     THIS PROGRAM CALCULATES A STIFFNESS MATRIX AND A DIAGONALIZED SKIP0040
C     CONSISTENT MASS MATRIX BY THE FINITE ELEMENT DISPLACEMENT   SKIP0050
C     FORMULATION METHOD USING NUMERICALLY INTEGRATED ISOPARAMETRIC SKIP0060
C     ELEMENTS.  A STODOLA ITERATION FOLLOWED BY AN               SKIP0070
C     EVALUATION OF THE RAYLEIGH QUOTENT IS THEN USED TO          SKIP0080
C     CALCULATE THE LOWEST NATURAL VIBRATIONAL FREQUENCY.         SKIP0090
C                                                                 SKIP0100
C        LT H. L. CREGO, USN, JUNE 1975                           SKIP0110
C*****************************************************************SKIP0120
C                                                                 SKIP0130
C                              NAT1-3D                            SKIP0140
C                                                                 SKIP0150
C                                                                 SKIP0160
C     TITLE CARD (10A8)  -  ANY PROBLEM TITLE IN COLUMNS 2-80     SKIP0170
C                                                                 SKIP0180
C                                                                 SKIP0190
C                                                                 SKIP0200
C     PROBLEM DATA (4I10)   - -  ONE CARD                         SKIP0210
C                                                                 SKIP0220
C          COLUMNS                                                SKIP0230
C            1-10       NUMBER OF NODES                    NNODE  SKIP0240
C           11-20       NUMBER OF ELEMENTS                 NEL    SKIP0250
C           21-30       NUMBER OF MATERIALS      MAX =  10 NMAT   SKIP0260
C           31-40       NUMBER OF NODES WITH               NNBC   SKIP0270
C                       BOUNDARY CONDITIONS      MAX =  30 NNBC   SKIP0280
C           41-50       NUMBER OF NODES PER ELEMENT        NPEL   SKIP0290
C                       8  LINEAR ELEMENTS                        SKIP0300
C                       20 QUADRATIC ELEMENTS                     SKIP0310
C           51-60       NUMBER OF GAUSS POINTS             NGP    SKIP0320
C                       2, 3, 4, 5, OR 6                          SKIP0330
C                                                                 SKIP0340
C                                                                 SKIP0350
C                                                                 SKIP0360
C     MATERIAL PROPERTIES - 1 CARD FOR EACH MATERIAL  (I10,3F10.2)SKIP0370
C                                                                 SKIP0380
C          COLUMNS                                                SKIP0390
C            1-10       MATERIAL IDENTIFICATION NUMBER           SKIP0400
C           11-20       YOUNG'S MODULUS   ( LB / IN SQ )         SKIP0410
C           21-30       POISSON'S RATIO                          SKIP0420
C           31-40       SPECIFIC WEIGHT  ( LB / CU IN )          SKIP0430
C                                                                 SKIP0440
C                                                                 SKIP0450
```

```
C                                                                       SKIP0460
C     NODE CARDS                                                        SKIP0470
C                    ---- CARDS PUNCHED BY TRIMEG ----                  SKIP0480
C                                                                       SKIP0490
C          COLUMNS                                                      SKIP0500
C            7-16        NODE NUMBER                                    SKIP0510
C           17-31        X COORDINATE                                   SKIP0520
C           32-46        Y COORDINATE                                   SKIP0530
C           47-61        Z COORDINATE                                   SKIP0540
C                                                                       SKIP0550
C                                                                       SKIP0560
C                                                                       SKIP0570
C     CONNECTIVITY MATRIX  -  2 CARDS PER ELEMENT  (10I5/11I5)          SKIP0580
C                                                                       SKIP0590
C                         ---- CARDS PUNCHED BY TRIMEG ----             SKIP0600
C                                                                       SKIP0610
C          COULMN                                                       SKIP0620
C            WIDTH                                                      SKIP0630
C              5          20 COLUMNS CONTAINING ELEMENT CONNECTIVITY     SKIP0640
C                          1 COLUMN DENOTING MATERIAL TYPE               SKIP0650
C                                                                       SKIP0660
C                                                                       SKIP0670
C                                                                       SKIP0680
C     BOUNDARY CONDITION CARDS  - 2 CARDS MAX  (15I5/15I5)              SKIP0690
C                                                                       SKIP0700
C          COLUMN                                                       SKIP0710
C           16-20          NODE NUMBER                                  SKIP0720
C              25          1  IF X-DIRECTION MOTION TO BE FIXED         SKIP0730
C              30          2  IF Y-DIRECTION MOTION TO BE FIXED         SKIP0740
C              35          3  IF Z-DIRECTION MOTION TO BE FIXED         SKIP0750
C                                                                       SKIP0760
C                                                                       SKIP0770
C***********************************************************************SKIP0780
C***********************************************************************SKIP0790
C     PRIOR TO EMPLOYING NAT1-3D THE USER MUST INSERT THE DIMENSIONING  SKIP0791
C     ARGUMENTS OF THE FOLLOWING LABELED COMMON STATEMENTS - - - -      SKIP0792
C              /BIGS/  /ENTER/  /ITER/  /SOL/                           SKIP0793
C     THE VALUES TO BE USED MAY BE DETERMINED AS INDICATED IN THIS      SKIP0794
C     LISTING.  THE DIMENSION OF THE VECTOR FMODE IN THE MAIN ROUTINE   SKIP0795
C     MUST ALSO BE SUPPLIED AS INDICATED.                               SKIP0796
C***********************************************************************SKIP0797
C***********************************************************************SKIP0798
C***********************************************************************SKIP0799
```

```
      IMPLICIT REAL*8(A-H,O-Z)                                          SKIP0800
C                                                                       SKIP0810
      COMMON /BIGS/BGK(NEQ,NBAND),ENN(3*NPEL,3*NPEL),ENW(3*NPEL,3*NPEL),BGM(NEQ)
C                                                                       SKIP0830
      COMMON /ITER/EMODE(NEQ),FACT                                      SKIP0850
C                                                                       SKIP0870
      COMMON /SOL/B(NEQ)                                                SKIP0880
C                                                                       SKIP0890
      COMMON /SING/NEL,NBAND,NEQ,NPEL,NGP                               SKIP0900
C                                                                       SKIP0910
      COMMON /FIX/ NBC(30),TITLE(10),NNBC                               SKIP0930
C                                                                       SKIP0940
      DIMENSION FMODE(NEQ)                                              SKIP0950
C                                                                       SKIP0960
      ZRO = 0.0D0                                                       SKIP0970
    1 CALL INPUT                                                        SKIP0980
      DO 10 I=1,NEQ                                                     SKIP0990
      DO 10 J=1,NBAND                                                   SKIP1000
   10 BGK(I,J) = ZRO                                                    SKIP1010
C                                                                       SKIP1020
C        COMPUTE THE STIFFNESS AND MASS MATRICIES                       SKIP1030
C                                                                       SKIP1040
      CALL MERGE                                                        SKIP1050
C                                                                       SKIP1060
C        SET-UP FOR THE STODLA ITERATION                                SKIP1070
C                                                                       SKIP1080
      CALL FASTEN                                                       SKIP1090
      FACT = 0.0D0                                                      SKIP1100
      ITER = 1                                                          SKIP1110
      ACC = 1.0D-6                                                      SKIP1120
      APZRO = 1.0D-6                                                    SKIP1130
C                                                                       SKIP1140
C        DECOMPOSE THE STIFFNESS MATRIX                                 SKIP1150
C                                                                       SKIP1160
      CALL LDLT(NEQ,NBAND)                                              SKIP1170
C                                                                       SKIP1180
C        THE ITERATION SECTION                                          SKIP1190
C        SET INITIAL GUESS AT 1                                         SKIP1200
C                                                                       SKIP1210
      DO 38 I=1,NEQ                                                     SKIP1220
   38 FMODE(I)=1.0D0                                                    SKIP1230
   39 TEST = FACT                                                       SKIP1240
      ITER = ITER + 1                                                   SKIP1250
      DO 40 I=1,NEQ                                                     SKIP1260
   40 EMODE(I)=BGM(I)*FMODE(I)                                          SKIP1270
      DO 41 I=1,NEQ                                                     SKIP1280
   41 B(I)=EMODE(I)                                                     SKIP1290
      CALL SOLV(NEQ,NBAND)                                              SKIP1300
```

```
      DO 43 I=1,NEQ                                                    SKIP1280
   43 EMODE(I)=B(I)                                                    SKIP1290
      CALL NORM                                                        SKIP1300
      DO 44 I=1,NEQ                                                    SKIP1310
   44 FMODE(I)=EMODE(I)                                                SKIP1320
      IF(ITER.EQ.1) GO TO 39                                           SKIP1330
      DELL = TEST-FACT                                                 SKIP1340
      IF(DABS(DELL).GT.ACC) GO TO 39                                   SKIP1350
      WRITE(6,400) ITER                                                SKIP1360
  400 FORMAT( ///,T25,' AFTER',I4,' ITERATIONS THE FIRST MODE VECTOR HASSKIP1370
     1 CONVERGED TO')                                                  SKIP1380
      WRITE(6,401) (I,EMODE(I),I=1,NEQ)                                SKIP1390
  401 FORMAT( //,44(/'   ',10(I4,F9.6)))                               SKIP1400
C                                                                      SKIP1410
CC         THE RAYLEIGH CALCULATION                                    SKIP1420
C                                                                      SKIP1430
      DO 50 I=1,NEQ                                                    SKIP1440
      FMODE(I)=EMODE(I)                                                SKIP1450
   50 B(I)=EMODE(I)                                                    SKIP1460
      DO 51 I=1,NEQ                                                    SKIP1470
   51 EMODE(I)=ZRO                                                     SKIP1480
      DENOM = ZRO                                                      SKIP1490
      DO 52 I=1,NEQ                                                    SKIP1500
   52 DENOM = DENOM+FMODE(I)*FMODE(I)*BGM(I)                           SKIP1510
      DO 60 I=1,NEQ                                                    SKIP1520
   60 FMODE(I)=BGK(I,1)                                                SKIP1530
      DO 61 I=1,NEQ                                                    SKIP1540
   61 BGK(I,1)=1.0D0                                                   SKIP1550
      DO 62 I=1,NEQ                                                    SKIP1560
      EMODE(I) = ZRO                                                   SKIP1570
      K=0                                                              SKIP1580
      DO 62 J=I,NEQ                                                    SKIP1590
      K=K+1                                                            SKIP1600
      IF(K.GT.NBAND) GO TO 62                                          SKIP1610
      EMODE(I)=EMODE(I)+B(J)*BGK(I,K)                                  SKIP1620
   62 CONTINUE                                                         SKIP1630
      TOP = 0.0D0                                                      SKIP1640
      DO 63 I=1,NEQ                                                    SKIP1650
   63 TOP = TOP+FMODE(I)*EMODE(I)*EMODE(I)                             SKIP1660
      WRITE(6,600) TOP,DENOM                                           SKIP1670
  600  FORMAT( ///,T25,' TOP = ', G25.16,10X,' DENOM = ',G25.16)       SKIP1680
      WS = TOP*3.8706D2/DENOM                                          SKIP1690
      W = DSQRT(WS)                                                    SKIP1700
      WRITE(6,601) W                                                   SKIP1710
  601 FORMAT( ///,T25,' THE VALUE OF OMEGA FROM THE RAYLEIGH QUOTENT IS'SKIP1720
     1,G25.16)                                                         SKIP1730
      GO TO 1                                                          SKIP1740
      END                                                              SKIP1750
```

```
C                                                                       INPU0010
      SUBROUTINE INPUT                                                  INPU0020
      IMPLICIT REAL*8(A-H,O-Z)                                          INPU0030
C                                                                       INPJ0040
      COMMON /ENTER/COARD(NNODE,3),NCONN(NEL,NPEL+1)
C                                                                       INPU0060
      COMMON /ELAS/ D(6,6),ELCON(10,3)                                  INPU0070
C                                                                       INPU0080
      COMMON /FIX/ NBC(30,4),TITLE(10),NNBC                             INPU0090
C                                                                       INPU0100
      COMMON /SING/NEL,NBAND,NEQ,NPEL,NGP                               INPU0110
C                                                                       INPU0120
CC                                                                      INPU0130
CC        READ PROBLEM PARAMETERS                                       INPU0140
C                                                                       INPJ0150
      ZRO = 0.0D0                                                       INPU0160
      READ(5,100,END=9000) TITLE                                        INPJ0170
  100 FORMAT( 10A8)                                                     INPU0180
      WRITE(6,101) TITLE                                                INPU0190
  101 FORMAT( //,15X,10A8,//)                                           INPU0200
      READ (5,200) NNODE,NEL,NMAT,NNBC,NPEL,NGP                         INPU0210
  200 FORMAT ( 6I10)                                                    INPU0220
      WRITE(6,201) NNODE,NEL,NMAT,NNBC,NPEL,NGP                         INPJ0230
  201 FORMAT( 11X,'THE NUMBER OF NODES IS ',10X,I10,/,                  INPU0240
     111X, 'THE NUMBER OF ELEMENTS IS',7X,I10,/,                        INPU0250
     211X, 'THE NUMBER OF MATERIALS IS',6X,I10,/,                       INPJ0260
     311X, 'THE NUMBER OF BOUNDED NODES IS',2X,I10,/,                   INPJ0270
     411X, 'THE NUMBER OF NODES PER ELEMENT IS',I8,/,                   INPU0280
     511X, 'THE NUMBER OF GAUSS POINTS USED IS',I8,///)                 INPU0290
      NEQ = NNODE*3                                                     INPU0300
C                                                                       INPU0310
CC        ZERO THE DATA MATRICIES                                       INPU0320
C                                                                       INPU0330
      K = NPEL + 1                                                      INPU0340
      DO 10 I=1,NEL                                                     INPU0350
      DO 10 J=1,K                                                       INPU0360
   10 NCONN(I,J) = 0                                                    INPU0370
      DO 20 I=1,NNODE                                                   INPU0380
      DO 20 J=1,3                                                       INPJ0390
   20 COARD(I,J) = ZRO                                                  INPU0400
C                                                                       INPU0410
CC        READ MATERIAL PROPERTIES                                      INPU0420
C                                                                       INPU0430
      DO 30 I=1,NMAT                                                    INPU0440
   30 READ(5,300) IMAT,(ELCON(IMAT,J),J=1,3)                            INPU0450
  300 FORMAT( 1I10,3F10.2)                                              INPU0460
      WRITE(6,301)                                                      INPU0470
  301 FORMAT(//,' MATERIAL PROPERTIES',//,3X,' MAT.NO. YOUNG,S MOD.   POINPU0480
```

```
     1ISSON,S RAT.    DENSITY',//)                                          INPU0490
      DO 31 I=1,NMAT                                                       INPU0500
   31 WRITE(6,302) I,(ELCON(I,J),J=1,3)                                    INPU0510
  302 FORMAT( 6X,I3,1X,4G15.6)                                             INPU0520
C                                                                          INPU0530
CC         READ THE JOINT COORDINATES                                      INPU0540
C                                                                          INPU0550
      WRITE (6,401)                                                        INPU0560
  401 FORMAT( //,T25,'COORDINATES OF THE JOINTS',//,' JOINT NUMBER',8X,    INPU0570
     1'X COORDINATE',14X,'Y COORDINATE',11X,'Z COORDINATE',//)             INPU0580
      DO 40 I=1,NNODE                                                      INPU0590
      READ(5,400) IJT,(COARD(IJT,J),J=1,3)                                 INPU0600
  400 FORMAT( 6X,I10,3F15.5)                                               INPU0610
      WRITE(6,402)(IJT,COARD(IJT,J),J=1,3)                                 INPU0620
  402 FORMAT( 5X,I3,12X,G14.5,2(3X8G14.5))                                 INPU0630
   40 CONTINUE                                                             INPU0640
C                                                                          INPU0650
CC         READ ELEMENT CONNECTIVITY                                       INPU0660
CC         AND COMPUTE HALF BAND WIDTH                                     INPU0670
C                                                                          INPU0680
      WRITE(6,499)                                                         INPU0690
  499 FORMAT( //,10X,' THE CONNECTIVITY MATRIX',//)                        INPU0700
      K = NPEL + 1                                                         INPU0710
      DO 50 I=1,NEL                                                        INPU0720
      READ(5,500)(NCONN(I,J),J=1,K)                                        INPU0730
  500 FORMAT( 10I5,/11I5)                                                  INPU0740
   50 WRITE(6,501)(NCONN(I,J),J=1,K)                                       INPU0750
  501 FORMAT (10X,21I5)                                                    INPU0760
      II = NPEL - 1                                                        INPU0770
      NBAND = 0                                                            INPU0780
      DO 53 I=1,NEL                                                        INPU0790
      DO 51 K=1,NPEL                                                       INPU0800
   51 NBC(K,1) = NCONN(I,K)                                                INPU0810
      DO 52 J=1,II                                                         INPU0820
      JK = J+1                                                             INPU0830
      DO 52 K=JK,NPEL                                                      INPU0840
   52 NBAND = MAXO(NBAND,IABS(NBC(J,1)-NBC(K,1)))                          INPU0850
   53 CONTINUE                                                             INPU0860
      NBAND = (NBAND+1)*3                                                  INPU0870
      WRITE (6,502) NBAND                                                  INPU0880
  502 FORMAT( ///,11X,' SYSTEM HALF BAND WIDTH = ',I5,///)                 INPU0890
C                                                                          INPU0900
CC         READ BOUNDARY CONDITION DATA                                    INPU0910
C                                                                          INPU0920
      DO 60 I=1,NNBC                                                       INPU0930
   60 READ(5,601)(NBC(I,J),J=1,4)                                          INPU0940
  601 FORMAT( 15X,4I5)                                                     INPU0950
      WRITE(6,602)                                                         INPU0960
```

```
  602 FORMAT( 10X,'THE NODES WITH BOUNDARY CONDITIONS ARE',//10X,'NODE NINPU0970
     1UMBER',5X,'COORDINATE DIRECTIONS FIXED',//)                        INPU0980
      DO 61 I=1,NNBC                                                     INPU0990
   61 WRITE(6,603)(NBC(I,J),J=1,4)                                       INPU1000
  603 FORMAT( 14X,I5,11X,3(3X,I1))                                       INPU1010
      RETURN                                                             INPU1020
 9000 STOP                                                               INPU1030
      END                                                                INPU1040
C                                                                        MERG0010
      SUBROUTINE MERGE                                                   MERG0020
      IMPLICIT REAL*8(A-H,O-Z)                                           MERG0030
C                                                                        MERG0040
C     COMMON /BIGS/BGK(NEQ,NBAND),ENN(3*NPEL,3*NPEL),ENW(3*NPEL,3*NPEL),BGM(NEQ)
C                                                                        MERG0060
      COMMON /ENTER/COARD(NNODE,3),NCONN(NEL,NPEL+1)
C                                                                        MERG0080
      COMMON /ELAS/ D(6,6),ELCON(10,3)                                   MERG0090
C                                                                        MERG0100
      COMMON /SING/NEL,NBAND,NEQ,NPEL,NGP                                MERG0110
C                                                                        MERG0120
      COMMON /DRONE/ AJI(3,3),AJ(3,3),DCOL(20,3),CORD(20,3),COL(20),DTJ  MERG0130
C                                                                        MERG0140
      DIMENSION LJT(20),LM(20)                                           MERG0150
C                                                                        MERG0160
      ZRO = 0.0D0                                                        MERG0170
C                                                                        MERG0180
CC                 FIRST FORM THE MASS MATRIX                            MERG0190
C                                                                        MERG0200
      DO 50 IK=1,NEL                                                     MERG0210
      DO 30 NN=1,NPEL                                                    MERG0220
      LJT(NN)=NCONN(IK,NN)                                               MERG0230
   30 LM(NN)=(NCONN(IK,NN)-1)*3                                          MERG0240
      DO 40 I1=1,NPEL                                                    MERG0250
      I2=LJT(I1)                                                         MERG0260
      DO 40 J1=1,3                                                       MERG0270
   40 CORD(I1,J1) = COARD(I2,J1)                                         MERG0280
      L = NPEL + 1                                                       MERG0290
      MATN = NCONN(IK,L)                                                 MERG0300
      RHO = ELCON(MATN,3)                                                MERG0310
      CALL QUAD (1,NPEL,NGP)                                             MERG0320
      DO 50 I=1,NPEL                                                     MERG0330
      DO 50 J=1,NPEL                                                     MERG0340
      DO 50 K=1,3                                                        MERG0350
      II = LM(I) +K                                                      MERG0360
      KK = 3*(I-1) + K                                                   MERG0370
      DO 50 L=1,3                                                        MERG0380
      JJ = LM(J) + L - II + 1                                            MERG0390
      IF(JJ.LE.0) GO TO 50                                               MERG0400
```

```
      LL = 3*(J-1)+L                                                    MERG0410
      BGK(II,JJ) = BGK(II,JJ) + ENW(KK,LL)*RHO                          MERG0420
   50 CONTINUE                                                          MERG0430
      SUM = ZRO                                                         MERG0440
      DO 60 I=1,NEQ                                                     MERG0450
      BGM(I) = BGK(I,1)                                                 MERG0460
   60 SUM = SUM + BGM(I)                                                MERG0470
      SUM = SUM/3.0D0                                                   MERG0480
      WRITE (6,65) SUM                                                  MERG0490
   65 FORMAT( ///,' WEIGHT CHECK ***  TOTAL WEIGHT = ',G15.6,///)       MERG0500
C                                                                       MERG0510
CC              NOW FORM THE STIFFNESS MATRIX                           MERG0520
C                                                                       MERG0530
      DO 70 I=1,NEQ                                                     MERG0540
      DO 70 J=1,NBAND                                                   MERG0550
   70 BGK(I,J) = ZRO                                                    MERG0560
      DO 100 IK=1,NEL                                                   MERG0570
      DO 80 NN=1,NPEL                                                   MERG0580
      LJT(NN) = NCONN(IK,NN)                                            MERG0590
   80 LM(NN) = (NCONN (IK,NN)-1)*3                                      MERG0600
      DO 90 I1=1,NPEL                                                   MERG0610
      I2=LJT(I1)                                                        MERG0620
      DO 90 J1=1,3                                                      MERG0630
   90 CORD(I1,J1) = COARD(I2,J1)                                        MERG0640
      L = NPEL + 1                                                      MERG0650
      MATN = NCONN(IK,L)                                                MERG0660
      E = ELCON(MATN,1)                                                 MERG0670
      PR = ELCON(MATN,2)                                                MERG0680
      CALL ELASM(E,PR)                                                  MERG0690
      CALL QUAD (2,NPEL,NGP)                                            MERG0700
      DO 100 I=1,NPEL                                                   MERG0710
      DO 100 J=1,NPEL                                                   MERG0720
      DO 100 K=1,3                                                      MERG0730
      II = LM(I) + K                                                    MERG0740
      KK = 3*(I-1) + K                                                  MERG0750
      DO 100 L=1,3                                                      MERG0760
      JJ = LM(J) +L-II+1                                                MERG0770
      IF(JJ.LE.0) GO TO 100                                             MERG0780
      LL = 3*(J-1)+L                                                    MERG0790
      BGK(II,JJ) = BGK(II,JJ) + ENW(KK,LL)                              MERG0800
  100 CONTINUE                                                          MERG0810
      RETURN                                                            MERG0820
      END                                                               MERG0830
C                                                                       MERG0840
C                                                                       ELAS0010
      SUBROUTINE ELASM (E,PR)                                           ELAS0020
      IMPLICIT REAL*8(A-H,O-Z)                                          ELAS0030
C                                                                       ELAS0040
```

```
      COMMON /ELAS/ D(6,6),ELCON(10,3)                                          ELAS0050
C                                                                               ELAS0060
CC    THIS SUBROUTINE FORMS THE CONSTITUTIVE ELASTICITY MATRIX -D-              ELAS0070
C                                                                               ELAS0080
      ZRO = 0.0D0                                                               ELAS0090
      DO 10 J=1,6                                                               ELAS0100
      DO 10 K=1,6                                                               ELAS0110
   10 D(J,K) = ZRO                                                              ELAS0120
      ER = E/((1.0D0 + PR)*(1.0D0-2.0D0*PR))                                    ELAS0130
      EK = ER*(1.0D0-PR)                                                        ELAS0140
      EL = ER*PR                                                                ELAS0150
      D(1,1) = EK                                                               ELAS0160
      D(1,2) = EL                                                               ELAS0170
      D(1,3) = EL                                                               ELAS0180
      D(2,1) = EL                                                               ELAS0190
      D(2,2) = EK                                                               ELAS0200
      D(2,3) = EL                                                               ELAS0210
      D(3,1) = EL                                                               ELAS0220
      D(3,2) = EL                                                               ELAS0230
      D(3,3) = EK                                                               ELAS0240
      D(4,4) = ER*(1.0D0-2.0D0*PR)/2.0D0                                        ELAS0250
      D(5,5) = D(4,4)                                                           ELAS0260
      D(6,6) = D(4,4)                                                           ELAS0270
      RETURN                                                                    ELAS0280
      END                                                                       ELAS0290
C                                                                               ELAS0300
CC                                                                              QUAD0010
      SUBROUTINE QUAD (ITYPE,NPEL,NGP)                                          QUAD0020
      IMPLICIT REAL*8(A-H,O-Z)                                                  QUAD0030
C                                                                               QUAD0040
CC    COMMON /BIGS/BGK(NEQ,NBAND),ENN(3*NPEL,3*NPEL),ENW(3*NPEL,3*NPEL),BGM(NEQ)
C                                                                               QUAD0060
      DIMENSION GP(6,5),WF(6,5)                                                 QUAD0070
C                                                                               QUAD0080
      DATA GP/.57735026918963D0,-.57735026918963D0,0.0D0,0.0D0,0.0D0,0.DQUAD0090
     10,.774596669241480D0, C.0D0,-.774596669241480D0,0.0D0,0.0D0,0.0D0,.86QUAD0100
     211363115940 5D0,0.33998104358486D0,-.33998104358486D0,-.861136311 59QUAD0110
     3405D0,0.0D0,0.0D0,.90617984593866D0,.53846931010568D0,0.0D0,-.5384QUAD0120
     469310105 68D0,-.90617984593866D0, 0.0D0,.93246951420315D0,.66120938QUAD0130
     5646626D0,.23861918608317D0,-.23861918608317D0,-.661209386466 26D0,-QUAD0140
     6.93246951420315D0/                                                        QUAD0150
C                                                                               QUAD0160
      DATA WF/1.0D0,1.0D0,0.0D0,0.0D0,0.0D0,0.0D0,.555555555555 56D0,.888QUAD0170
     1888888888 9D0,.555555555555 56D0,0.0D0,0.0D0,0.0D0,.34785484513745DQUAD0180
     220,.65214515486255D0,.65214515486255D0,.34785484513745D0,0.0D0,0.0DQUAD0190
     330,.23692688505619D0,.47862867049937D0,.56888888888889D0,.4786286 7QUAD0200
     449937D0,.23692688505619D0,0.0D0,.17132449237917D0,.36076157304814DQUAD0210
     550,.46791393457269D0,.46791393457269D0,.36076157304814D0,.17132449 2QUAD0220
```

```
     637917D0/                                                          QUAD0230
C                                                                       QUAD0240
      ZRO = 0.0D0                                                       QUAD0250
      NW = NPEL*3                                                       QUAD0260
      IDGP = NGP - 1                                                    QUAD0270
      DO 10 I=1,NW                                                      QUAD0280
      DO 10 J=1,NW                                                      QUAD0290
   10 ENW (I,J) = ZRO                                                   QUAD0300
      GO TO (1000,2000), ITYPE                                          QUAD0310
 1000 CONTINUE                                                          QUAD0320
C                                                                       QUAD0330
CC              THIS SECTION USED FOR BGM                               QUAD0340
CC                                                                      QUAD0350
CC                                                                      QUAD0360
CC              COMPUTATION OF PRODUCT OF GAUSS WEIGHTS                 QUAD0370
CC                        AND                                           QUAD0380
CC           SELECTION OF POINTS OF INTEGRATION                         QUAD0390
C                                                                       QUAD0400
      DO 30 I=1,NGP                                                     QUAD0410
      XI = GP(I,IDGP)                                                   QUAD0420
      WX = WF(I,IDGP)                                                   QUAD0430
      DO 30 J=1,NGP                                                     QUAD0440
      ETA = GP(J,IDGP)                                                  QUAD0450
      WY = WF(J,IDGP)                                                   QUAD0460
      DO 30 K=1,NGP                                                     QUAD0470
      ZETA = GP(K,IDGP)                                                 QUAD0480
      WZ = WF(K,IDGP)                                                   QUAD0490
      WFACT = WX*WY*WZ                                                  QUAD0500
      CALL FORMM(XI,ETA,ZETA,NPEL)                                      QUAD0510
      DO 20 L=1,NW                                                      QUAD0520
      DO 20 M=1,NW                                                      QUAD0530
   20 ENW(L,M) = ENW(L,M) + WFACT*ENN(L,M)                              QUAD0540
   30 CONTINUE                                                          QUAD0550
C                                                                       QUAD0560
CC              NOW DIAGONALIZE THE ELEMENT MASS MATRIX                 QUAD0570
C                                                                       QUAD0580
      DO 50 I=1,NW                                                      QUAD0590
      RSUM = ZRO                                                        QUAD0600
      DO 40 J=1,NW                                                      QUAD0610
      RSUM = RSUM + ENW(I,J)                                            QUAD0620
   40 ENW(I,J) = ZRO                                                    QUAD0630
   50 ENW(I,I) = RSUM/3.0D0                                             QUAD0640
      GO TO 3000                                                        QUAD0650
 2000 CONTINUE                                                          QUAD0660
C                                                                       QUAD0670
CC              THIS SECTION USED FOR BGK                               QUAD0680
C                                                                       QUAD0690
      DO 70 I=1,NGP                                                     QUAD0700
```

```
      XI = GP(I,IDGP)                                                    QUAD0710
      WX = WF(I,IDGP)                                                    QUAD0720
      DO 70 J=1,NGP                                                      QUAD0730
      ETA = GP(J,IDGP)                                                   QUAD0740
      WY = WF(J,IDGP)                                                    QUAD0750
      DO 70 K=1,NGP                                                      QUAD0760
      ZETA = GP(K,IDGP)                                                  QUAD0770
      WZ = WF(K,IDGP)                                                    QUAD0780
      WFACT = WX*WY*WZ                                                   QUAD0790
      CALL FORMK(XI,ETA,ZETA,NPEL)                                       QUAD0800
      DO 60 L=1,NW                                                       QUAD0810
      DO 60 M=1,NW                                                       QUAD0820
   60 ENW(L,M) = ENW(L,M) + WFACT*ENN(L,M)                               QUAD0830
   70 CONTINUE                                                           QUAD0840
 3000 CONTINUE                                                           QUAD0850
      RETURN                                                             QUAD0860
      END                                                                QUAD0870
C                                                                        FORM0010
      SUBROUTINE FORMM(XI,ETA,ZETA,NPEL)                                 FORM0020
      IMPLICIT REAL*8(A-H,O-Z)                                           FORM0030
C                                                                        FORM0040
CC    COMMON /BIGS/BGK(NEQ,NBAND),ENN(3*NPEL,3*NPEL),ENW(3*NPEL,3*NPEL),BGM(NEQ)
C                                                                        FORM0060
      COMMON /DRONE/ AJI(3,3),AJ(3,3),DCOL(20,3),CORD(20,3),COL(20),DTJ  FORM0070
C                                                                        FORM0080
      COMMON /ELAS/ D(6,6),ELCON(10,3)                                   FORM0090
C                                                                        FORM0100
      DIMENSION B1(60,6),B(6,60),TEMP(60)                                FORM0110
C                                                                        FORM0120
CC                THIS SECTION IS USED TO FORM BGM                       FORM0130
C                                                                        FORM0140
      NW = NPEL*3                                                        FORM0150
      ZRO = 0.0D0                                                        FORM0160
      DO 10 I=1,NW                                                       FORM0170
      TEMP(I) = ZRO                                                      FORM0180
      DO 10 J=1,NW                                                       FORM0190
   10 ENN(I,J) = ZRO                                                     FORM0200
C                                                                        FORM0210
CC                COMPUTE SHAPE FUNCTION PRODUCTS                        FORM0220
C                                                                        FORM0230
      CALL SHAPE(XI,ETA,ZETA,NPEL)                                       FORM0240
      CALL JACOB(NPEL)                                                   FORM0250
      K = 0                                                              FORM0260
      DO 20 J=1,NPEL                                                     FORM0270
      DO 20 I=1,3                                                        FORM0280
      K=K+1                                                              FORM0290
   20 TEMP(K) = COL(J)                                                   FORM0300
      DO 30 I=1,NW                                                       FORM0310
```

```
      A = TEMP(I) *DTJ                                                  FORM0320
      DO 30 J=1,NW                                                      FORM0330
   30 ENN(I,J) = A*TEMP(J)                                              FORM0340
      RETURN                                                            FORM0350
C                                                                       FORM0360
C*********************************************************************FORM0370
C                                                                       FORM0380
CC              THIS SECTION USED TO FORM BGK                           FORM0390
C                                                                       FORM0400
      ENTRY FORMK(XI,ETA,ZETA,NPEL)                                     FORM0410
      NW = NPEL*3                                                       FORM0420
      ZRO = 0.0D0                                                       FORM0430
      CALL SHAPE(XI,ETA,ZETA,NPEL)                                      FORM0440
      CALL JACOB(NPEL)                                                  FORM0450
      DO 21 I=1,NPEL                                                    FORM0460
      DNX = DCOL(I,1)                                                   FORM0470
      DNY = DCOL(I,2)                                                   FORM0480
      DNZ = DCOL(I,3)                                                   FORM0490
      DCOL(I,1) = AJI(1,1)*DNX+AJI(1,2)*DNY+AJI(1,3)*DNZ                FORM0500
      DCOL(I,2) = AJI(2,1)*DNX+AJI(2,2)*DNY+AJI(2,3)*DNZ                FORM0510
   21 DCOL(I,3) = AJI(3,1)*DNX+AJI(3,2)*DNY+AJI(3,3)*DNZ                FORM0520
      DO 31 I=1,6                                                       FORM0530
      DO 31 J=1,NW                                                      FORM0540
      B1(J,I) = ZRO                                                     FORM0550
   31 B(I,J) = ZRO                                                      FORM0560
C                                                                       FORM0570
CC              FORM N* MATRIX                                          FORM0580
C                                                                       FORM0590
      DO 40 I=1,NPEL                                                    FORM0600
      I1=3*I-2                                                          FORM0610
      I2=I1+1                                                           FORM0620
      I3=I2+1                                                           FORM0630
      B(1,I1) = DCOL(I,1)                                               FORM0640
      B(2,I2) = DCOL(I,2)                                               FORM0650
      B(3,I3) = DCOL(I,3)                                               FORM0660
      B(4,I1) = DCOL(I,2)                                               FORM0670
      B(4,I2) = DCOL(I,1)                                               FORM0680
      B(5,I2) = DCOL(I,3)                                               FORM0690
      B(5,I3) = DCOL(I,2)                                               FORM0700
      B(6,I3) = DCOL(I,1)                                               FORM0710
   40 B(6,I1) = DCOL(I,3)                                               FORM0720
      DO 50 I=1,NW                                                      FORM0730
      DO 50 J=1,6                                                       FORM0740
      DO 50 K=1,6                                                       FORM0750
   50 B1(I,J)=B1(I,J)+B(K,I)*D(K,J)                                     FORM0760
      DO 60 I=1,NW                                                      FORM0770
      DO 60 J=1,NW                                                      FORM0780
      ENN(I,J) = ZRO                                                    FORM0790
```

```
      DO 60 K=1,6                                                       FORM0800
   60 ENN(I,J)=ENN(I,J)+B1(I,K)*B(K,J)                                  FORM0810
      DO 80 I=1,NW                                                      FORM0820
      DO 80 J=I,NW                                                      FORM0830
      ENN(I,J)=ENN(I,J)*DTJ                                             FORM0840
   80 ENN(J,I)=ENN(I,J)                                                 FORM0850
      RETURN                                                            FORM0860
      END                                                               FORM0870
C                                                                       FORM0880
C                                                                       SHAP0010
      SUBROUTINE SHAPE (XI,ETA,ZETA,NPEL)                               SHAP0020
      IMPLICIT REAL*8(A-H,O-Z)                                          SHAP0030
C                                                                       SHAP0040
      COMMON /DRONE/ AJI(3,3),AJ(3,3),DCOL(20,3),CORD(20,3),COL(20),DTJ SHAP0050
C                                                                       SHAP0060
      DIMENSION CORDX(4),CORDY(4),CORDZ(4)                              SHAP0070
C                                                                       SHAP0080
      DATA CORDX/1.0D0,-1.0D0,-1.0D0,1.0D0/                             SHAP0090
      DATA CORDY/1.0D0,1.0D0,-1.0D0,-1.0D0/                             SHAP0100
      DATA CORDZ/1.0D0,-1.0D0,1.0D0,-1.0D0/                             SHAP0110
C                                                                       SHAP0120
C                                                                       SHAP0130
      FL(XI,ETA,ZETA,X1,Y1,Z1)=(1.0D0+XI*X1)*(1.0D0+ETA*Y1)*(1.0D0+ZETA*SHAP0140
     1Z1)/8.0D0                                                         SHAP0150
C                                                                       SHAP0160
      DFL(ETA,ZETA,X1,Y1,Z1)=X1*(1.0D0+ETA*Y1)*(1.0D0+ZETA*Z1)/8.0D0    SHAP0170
C                                                                       SHAP0180
      FQC(XI,ETA,ZETA,X1,Y1,Z1)=(1.0D0+XI*X1)*(1.0D0+ETA*Y1)*(1.0D0+ZETASHAP0190
     1*Z1)*(XI*X1+ETA*Y1+ZETA*Z1-2.0D0)/8.0D0                           SHAP0200
C                                                                       SHAP0210
      FQM(XI,ETA,ZETA,Y1,Z1)=(1.0D0-XI*XI)*(1.0D0+ETA*Y1)*(1.0D0+ZETA*  SHAP0220
     1Z1)/4.0D0                                                         SHAP0230
C                                                                       SHAP0240
      DFQC(XI,ETA,ZETA,X1,Y1,Z1)=X1*(1.0D0+ETA*Y1)*(1.0D0+ZETA*Z1)*(2.0 SHAP0250
     1D0*XI*X1+ETA*Y1+ZETA*Z1-1.0D0)/8.0D0                              SHAP0260
C                                                                       SHAP0270
      DFQMX(XI,ETA,ZETA,Y1,Z1)=-XI*(1.0D0+ETA*Y1)*(1.0D0+ZETA*Z1)/2.0D0 SHAP0280
C                                                                       SHAP0290
      DFQMY(XI,ZETA,Y1,Z1)=Y1*(1.0D0-XI*XI)*(1.0D0+ZETA*Z1)/4.0D0       SHAP0300
C                                                                       SHAP0310
C                                                                       SHAP0320
      IGO = (NPEL-8)/12 + 1                                             SHAP0330
      GO TO (100,200), IGO                                              SHAP0340
C                                                                       SHAP0350
C        LINEAR SHAPR FUNCTIONS                                         SHAP0360
C                                                                       SHAP0370
  100 ICOL = 0                                                          SHAP0380
      DO 10 IZ=1,2                                                      SHAP0390
```

```
      DO 10 IX=1,4                                                      SHAP0400
      ICOL = ICOL + 1                                                   SHAP0410
      X1 = CORDX(IX)                                                    SHAP0420
      Y1 = CORDY(IX)                                                    SHAP0430
      Z1 = CORDZ(IZ)                                                    SHAP0440
      COL(ICOL) = FL(XI,ETA,ZETA,X1,Y1,Z1)                              SHAP0450
      DCOL(ICOL,1) = DFL(ETA,ZETA,X1,Y1,Z1)                             SHAP0460
      DCOL(ICOL,2) = DFL(ZETA,XI,Y1,Z1,X1)                              SHAP0470
   10 DCOL(ICOL,3) = DFL(XI,ETA,Z1,X1,Y1)                               SHAP0480
      GO TO 40                                                          SHAP0490
C                                                                       SHAP0500
CCCCCC   QUADRATIC SHAPE FUNCTIONS                                      SHAP0510
C                                                                       SHAP0520
C                                                                       SHAP0530
C           CORNER NODES                                                SHAP0540
C           1,3,5,7,13,15,17,19                                         SHAP0550
C                                                                       SHAP0560
  200 CONTINUE                                                          SHAP0570
      DO 20 IZ=1,2                                                      SHAP0580
      II=12*(IZ-1)+1                                                    SHAP0590
      IT=II+6                                                           SHAP0600
      IX=0                                                              SHAP0610
      DO 20 ICOL=II,IT,2                                                SHAP0620
      IX=IX+1                                                           SHAP0630
      Y1=CORDY(IX)                                                      SHAP0640
      X1=CORDX(IX)                                                      SHAP0650
      Z1=CORDZ(IZ)                                                      SHAP0660
      COL(ICOL)=FQC(XI,ETA,ZETA,X1,Y1,Z1)                               SHAP0670
      DCOL(ICOL,1)=DFQC(XI,ETA,ZETA,X1,Y1,Z1)                           SHAP0680
      DCOL(ICOL,2)=DFQC(ETA,ZETA,XI,Y1,Z1,X1)                           SHAP0690
   20 DCOL(ICOL,3)=DFQC(ZETA,XI,ETA,Z1,X1,Y1)                           SHAP0700
C                                                                       SHAP0710
C           MID SIDE NODES                                              SHAP0720
C           2,6,14,18,4,8,16,20, 9,10,11,12                             SHAP0730
C                                                                       SHAP0740
      IC9=8                                                             SHAP0750
      DO 30 IZ=1,2                                                      SHAP0760
      II=12*(IZ-1)+2                                                    SHAP0770
      IT=II+4                                                           SHAP0780
      DO 30 IC2=II,IT,4                                                 SHAP0790
      IC4=IC2+2                                                         SHAP0800
      IC9=IC9+1                                                         SHAP0810
      IN=IC9-8                                                          SHAP0820
      X1=CORDX(IN)                                                      SHAP0830
      Y1=CORDY(IN)                                                      SHAP0840
      Z1=CORDZ(IN)                                                      SHAP0850
      COL(IC2)=FQM(XI,ETA,ZETA,Z1,Y1)                                   SHAP0860
      COL(IC4)=FQM(ETA,ZETA,XI,Y1,-Z1)                                  SHAP0870
```

```
      COL(IC9)=FQM(ZETA,XI,ETA,X1,Y1)                                   SHAP0880
      DCOL(IC2,1)=DFQMX(XI,ETA,ZETA,Z1,Y1)                              SHAP0890
      DCOL(IC2,2)=DFQMY(XI,ZETA,Z1,Y1)                                  SHAP0900
      DCOL(IC2,3)=DFQMY(XI,ETA,Y1,Z1)                                   SHAP0910
      DCOL(IC4,1)=DFQMY(ETA,ZETA,-Z1,Y1)                                SHAP0920
      DCOL(IC4,2)=DFQMX(ETA,XI,ZETA,-Z1,Y1)                             SHAP0930
      DCOL(IC4,3)=DFQMY(ETA,XI,Y1,-Z1)                                  SHAP0940
      DCOL(IC9,1) = DFQMY(ZETA,ETA,X1,Y1)                               SHAP0950
      DCOL(IC9,2)=DFQMY(ZETA,XI,Y1,X1)                                  SHAP0960
   30 DCOL(IC9,3)=DFQMX(ZETA,XI,ETA,X1,Y1)                              SHAP0970
   40 RETURN                                                            SHAP0980
      END                                                               SHAP0990
C                                                                       SHAP1000
C                                                                       JAC00010
      SUBROUTINE JACOB(NPEL)                                            JAC00020
      IMPLICIT REAL*8(A-H,O-Z)                                          JAC00030
C                                                                       JAC00040
      COMMON /DRONE/ AJI(3,3),AJ(3,3),DCOL(20,3),CORD(20,3),COL(20),DTJ JAC00050
C                                                                       JAC00060
      DO 10 I=1,3                                                       JAC00070
      DO 10 J=1,3                                                       JAC00080
      AJ(I,J) = 0.0D0                                                   JAC00090
      DO 10 K=1,NPEL                                                    JAC00100
   10 AJ(I,J) = AJ(I,J)+DCOL(K,I)*CORD(K,J)                             JAC00110
C               COMPUTE THE DETERMINANT                                 JAC00120
      DTJ=AJ(1,1)*AJ(2,2)*AJ(3,3)+AJ(1,2)*AJ(2,3)*AJ(3,1)+AJ(1,3)*AJ(2,1JAC00130
     1)*AJ(3,2)-AJ(3,1)*AJ(2,2)*AJ(1,3)-AJ(3,2)*AJ(2,3)*AJ(1,1)-AJ(3,3)*JAC00140
     2AJ(2,1)*AJ(1,2)                                                   JAC00150
C                                                                       JAC00160
C               COMPUTE THE INVERSE                                     JAC00170
C                                                                       JAC00180
      AJI(1,1)=(AJ(2,2)*AJ(3,3)-AJ(3,2)*AJ(2,3))/DTJ                    JAC00190
      AJI(2,1)=-(AJ(2,1)*AJ(3,3)-AJ(3,1)*AJ(2,3))/DTJ                   JAC00200
      AJI(3,1)= (AJ(2,1)*AJ(3,2)-AJ(3,1)*AJ(2,2))/DTJ                   JAC00210
      AJI(1,2)= -(AJ(1,2)*AJ(3,3)-AJ(3,2)*AJ(1,3))/DTJ                  JAC00220
      AJI(2,2)= (AJ(1,1)*AJ(3,3)-AJ(3,1)*AJ(1,3))/DTJ                   JAC00230
      AJI(3,2) =-(AJ(1,1)*AJ(3,2)-AJ(3,1)*AJ(1,2))/DTJ                  JAC00240
      AJI(1,3)= (AJ(1,2)*AJ(2,3)-AJ(2,2)*AJ(1,3))/DTJ                   JAC00250
      AJI(2,3)= -(AJ(1,1)*AJ(2,3)-AJ(2,1)*AJ(1,3))/DTJ                  JAC00260
      AJI(3,3)= (AJ(1,1)*AJ(2,2)-AJ(2,1)*AJ(1,2))/DTJ                   JAC00270
      RETURN                                                            JAC00280
      END                                                               JAC00290
C                                                                       JAC00300
C                                                                       FAST0010
      SUBROUTINE FASTEN                                                 FAST0020
      IMPLICIT REAL*8(A-H,O-Z)                                          FAST0030
C                                                                       FAST0040
C     COMMON /BIGS/BGK(NEQ,NBAND),ENN(3*NPEL,3*NPEL),ENW(3*NPEL,3*NPEL),BGM(NEQ)
```

```
C                                                                       FAST0060
      COMMON /FIX/ NBC(30,4),TITLE(10),NNBC                             FAST0070
C                                                                       FAST0080
      WELD=1.0D10                                                       FAST0090
      DO 20 I=1,NNBC                                                    FAST0100
      DO 10 K=2,4                                                       FAST0110
      IFIX = NBC(I,K)                                                   FAST0120
      IF(IFIX.EQ.0) GO TO 10                                            FAST0130
      JTFIX = NBC(I,1)*3-3+IFIX                                         FAST0140
      BGK(JTFIX,1) = WELD*BGK(JTFIX,1)                                  FAST0150
   10 CONTINUE                                                          FAST0160
   20 CONTINUE                                                          FAST0170
      RETURN                                                            FAST0180
      END                                                               FAST0190
C                                                                       NORM0010
      SUBROUTINE NORM                                                   NORM0020
      IMPLICIT REAL*8(A-H,O-Z)                                          NORM0030
C     COMMON /ITER/EMODE(NEQ),FACT
      COMMON /SING/ NEL,NBAND,NEQ,NPEL,NGP                              NORM0050
      SUM = 0.0D0                                                       NORM0060
      DO 10 J=1,NEQ                                                     NORM0070
   10 SUM = SUM+EMODE(J)*EMODE(J)                                       NORM0080
      FACT=DSQRT(SUM)                                                   NORM0090
      DO 20 K=1,NEQ                                                     NORM0100
   20 EMODE(K)=EMODE(K)/FACT                                            NORM0110
      WS = 3.8706D2/FACT                                                NORM0120
      W = DSQRT(WS)                                                     NORM0130
      WRITE(6,100) W                                                    NORM0140
  100 FORMAT( //,10X,' STODOLA OMEGA = ',G25.16)                        NORM0150
      RETURN                                                            NORM0160
      END                                                               NORM0170
C                                                                       LDLT0010
      SUBROUTINE LDLT(N,M)                                              LDLT0020
      IMPLICIT REAL*8(A-H,O-Z)                                          LDLT0030
C                                                                       LDLT0040
C     COMMON /BIGS/BGK(NEQ,NBAND),ENN(3*NPEL,3*NPEL),ENW(3*NPEL,3*NPEL),BGM(NEQ)
C                                                                       LDLT0060
      AN = N                                                            LDLT0070
      ZRO=0.0D0                                                         LDLT0080
      SUM = ZRO                                                         LDLT0090
      DO 30 I=1,N                                                       LDLT0100
   30 SUM = SUM+BGK(I,1)                                                LDLT0110
      ABGN = SUM*1.0D20                                                 LDLT0120
      APZRO = ABGN*1.0D-30/AN                                           LDLT0130
      AVSN = APZRO/ABGN                                                 LDLT0140
      NM = N-1                                                          LDLT0150
      DO 300 I=1,NM                                                     LDLT0160
      DIAG = BGK(I,1)                                                   LDLT0170
```

```
      IF(DIAG.EQ.ZRO) GO TO 400                                          LDLT0180
      IF(DIAG.LE.APZRO) WRITE(6,2000) I                                  LDLT0190
      DO 100 J=2,M                                                       LDLT0200
      IF(DABS(BGK(I,J)).LE.AVSN) BGK(I,J)=ZRO                            LDLT0210
  100 BGK(I,J) = BGK(I,J)/DIAG                                           LDLT0220
      DO 300 J=2,M                                                       LDLT0230
      L = I+J-1                                                          LDLT0240
      IF(L.GT.N) GO TO 300                                               LDLT0250
      AA = BGK(I,J)*DIAG                                                 LDLT0260
      IF(AA.EQ.ZRO) GO TO 300                                            LDLT0270
      DO 200 K=J,M                                                       LDLT0280
      ML = I+K-J                                                         LDLT0290
  200 BGK(L,ML) = BGK(L,ML)-AA*BGK(I,K)                                  LDLT0300
  300 CONTINUE                                                           LDLT0310
      GO TO 5000                                                         LDLT0320
  400 WRITE(6,1000) I                                                    LDLT0330
 1000 FORMAT(//,5X,'MATRIX IS SINGULAR',II5)                             LDLT0340
      GO TO 5000                                                         LDLT0350
 2000 FORMAT(//,5X,'WARNING MATRIX IS NEARLY SINGULAR EXECUTION CONTI    LDLT0360
     1NUES ', II5,5X,' RESIDUAL ITERATION MAY YIELD ERRONEOUS RESULTS    LDLT0370
     2    '//)                                                          LDLT0380
 5000 CONTINUE                                                          LDLT0390
      RETURN                                                             LDLT0400
      END                                                                LDLT0410
C                                                                        SOLV0010
      SUBROUTINE SOLV(N,M)                                               SOLV0020
      IMPLICIT REAL*8(A-H,O-Z)                                           SOLV0030
C                                                                        SOLV0040
CC                                                                       SOLV0050
CC    COMMON /BIGS/BGK(NEQ,NBAND),ENN(3*NPEL,3*NPEL),ENW(3*NPEL,3*NPEL),BGM(NEQ)
CC                                                                       SOLV0070
CC    COMMON /SOL/B(NEQ)
C                                                                        SOLV0090
      NM=N-1                                                             SOLV0100
      DO 100 I=1,NM                                                      SOLV0110
      BI=B(I)                                                            SOLV0120
      DO 100 J=2,M                                                       SOLV0130
      L=I+J-1                                                            SOLV0140
      IF(L.GT.N) GO TO 100                                               SOLV0150
      B(L) = B(L)-BGK(I,J)*BI                                            SOLV0160
  100 CONTINUE                                                           SOLV0170
      DO 200 I=1,N                                                       SOLV0180
  200 B(I) = B(I)/BGK(I,1)                                               SOLV0190
      DO 300 L=2,N                                                       SOLV0200
      IR=N-L+1                                                           SOLV0210
      BIRP=B(IR+1)                                                       SOLV0220
      DO 300 J=2,M                                                       SOLV0230
      IRO=IR-J+2                                                         SOLV0240
```

```
      IF(IRO.LE.0) GO TO 300                                            SOLV0250
      B(IRO) = B(IRO)-BGK(IRO,J)*BIRP                                   SOLV0260
  300 CONTINUE                                                          SOLV0270
      RETURN                                                            SOLV0280
      END                                                               SOLV0290
```

APPENDIX C

MESH GENERATION USING TRIMEG

The purpose of this appendix is to give the user who wishes to use TRIMEG a few brief but very important suggestions that could save time and reduce frustration.

A. BACKGROUND

TRIMEG [Ref. 1] in an automatic mesh generator for use with three dimensional isoparametric elements. It is both quick and efficient and saves the user a lot of tedious, time consuming work by punching the bulk of the data cards required to solve the vibrational problem.

B. COMMENTS

When using TRIMEG there are a few subtle guides which must not be overlooked in order to obtain the most efficient discretization.

1. Orientation

Unless the user has a reasonably good idea of the purposes and functions of TRIMEG it is likely that he will not use NAT1-3D to the greatest advantage. He is likely to orient his super elements or provide for subsequent subdivision of super elements into a number of individual elements in a disadvantageous way so as to use up available storage while accommodating a smaller total number of elements than would have been possible by a more effective arrangement.

Accordingly he is urged to study Ref. 2 before undertaking a problem. In particular, referring to Figure 13, p. 29, Ref. 2, the following general remarks may be of assistance.

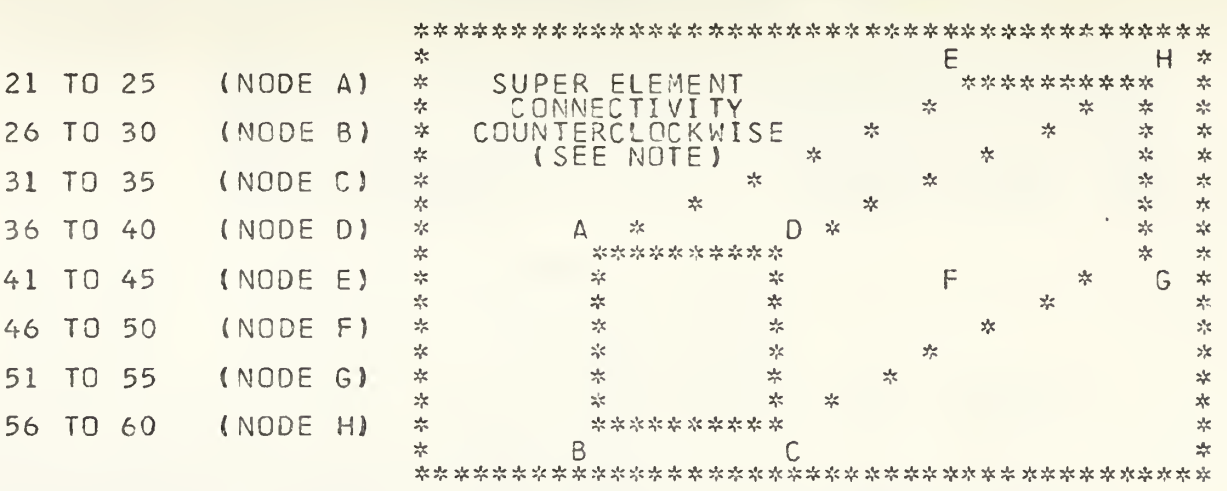
The super elements all have the same orientation given by xi, eta, and zeta directions. A super element is divided into ROW parts in the eta direction, into COL parts in the xi, direction, and into SLICE parts in the zeta direction. The body represented by the super elements has, let us say, a maximum of nxi super elements in the xi direction, a maximum of neta super elements in the eta direction, and a maximum of nzeta super elements in the zeta direction. The orientation of the super elements should be chosen so that the numerical products (ROW)(nxi), (COL)(neta), and (SLICE)(nzeta) are in ascending order. This gives minimum bandwidth corresponding to the total number of elements into which the body is finally subdivided.

2. Boundary Nodes

For configurations having curved boundaries it is possible and advisable to use 32 boundary nodes to describe the super element even when linear or quadratic shape functions are to be used. The use of the additional nodes allows TRIMEG to more accurately follow the curvature of the boundary of the object as it subdivides the super element.

```
C*************************************************************
C*                MESH GENERATING PROGRAM FOR 'TRISOP'       *
C*                     WITH QUADRATIC ELEMENTS               *
C*************************************************************
C*     CODED BY J.R.ADAMEK, MARCH 1973, NAVAL POST GRADUATE SCHOOL  *
C*************************************************************
C
C        THE NECESSARY INPUT FOR THIS PROGRAM IS AS FOLLOWS:
C
C  CARD 1 AND 2, TITLE, FORMAT (6A8) EACH CARD.
C
C     COL. 1 TO 48   BRIEF TITLE OF PROBLEM. USER'S NAME AND/OR BOX NO.
C                    MUST BE INCLUDED ON ONE OF THE CARDS IF CALCOMP
C                    OPTION INCLUDED. (SEE CARD 3)
C
C  CARD 3, MESH PARAMENTERS, FORMAT (3I5,3F5.0)
C
C     COL. 1 TO  5   (NSEL):  NUMBER OF SUPER ELEMENTS.
C
C          6 TO 10   (NPUNCH): IF THIS IS ZERO OR BLANK, NO CARDS WILL
C                    BE PUNCHED. IF NPUNCH IS DIFFERENT FROM ZERO A DECK
C                    FOR CONNECTIVITY AND X,Y & Z COORDINATES OF JOINTS
C                    WILL BE PUNCHED. NPUNCH BLANK OR ZERO SHOULD ALWAYS
C                    BE USED UNTIL ONE IS SATISFIED WITH THE MESH.
C
C         11 TO 15   (NPLOT):  IF THIS IS ZERO OR BLANK, NO PLOTS WILL
C                    BE MADE. IF NPLOT IS (1) A PLOT WILL BE OBTAINED ON
C                    THE CALCOMP PLOTTER.
C                    ALL PLOTS WILL BE PLOTTED IN FIRST QUADRANT ONLY.
C
C         THE FOLLOWING THREE INPUT VARIABLES ARE THE EULER ANGLES.
C         NOT REQUIRED UNLESS NPLOT = 1.
C
C         16 TO 20   (TETA):   ANGLE OF ROTATION ABOUT Z-AXIS.
C
C         21 TO 25   (ALPHA):  ANGLE OF ROTATION ABOUT LATEST X-AXIS.
C
C         26 TO 30   (BETA):   ANGLE OF ROTATION ABOUT LATEST Y-AXIS.
C
C  SUPER ELEMENT DECK, TOTAL OF NSEL CARDS, FORMAT (14I5), LOAD IN
C                      ASCENDING ORDER OF SUPER ELEMENT NUMBERS.
C
C     COL. 1 TO  5   (SEL NO.) SUPER ELEMENT IDENTIFICATION NUMBER.
C
C          6 TO 10   (ROW):    NUMBER OF ROWS IN SUPER ELEMENT.
C
C         11 TO 15   (COL):    NUMBER OF COLUMNS IN SUPER ELEMENT.
C
```

16 TO 20 (SLICE): NUMBER OF SLICES IN SUPER ELEMENT.



NOTE: NUMBERING OF SUPER-ELEMENT CORNER NODES IS ARBITRARY. HOWEVER, CORNER NODES WHICH ARE CONNECTED TO OTHER CORNER NODES, MUST HAVE THE SAME NUMBER.

61 TO 65 (TYPE): MATERIAL IDENTFICATION NUMBER FOR SUPER ELEMENT.(SEE 'TRISOP' INST)

66 TO 70 (NPTSB): NUMBER OF SUPER ELEMENT BOUNDARY NODES. NPTSB = 8,20 OR 32

SUPER ELEMENT BOUNDARY DECK, TOTAL OF NPTB CARDS, (3I5,3F20.0).

NPTB=SUM OF BOUNDARY NODES OF ALL SUPER ELEMENTS. ORDER IS COUNTER CLOCK-WISE ABOUT ZETA AXIS, STARTING AT NODE COMMON WITH (XI,ETA,ZETA)=(1,1,1). IF MORE THAN ONE SUPER ELEMENT, LOAD ITS BOUNDARY AFTER THAT OF FIRST SUPER ELEMENT IN SAME NODAL ORDER AS ABOVE. ORDER OF SUPER ELEMENTS SAME AS FOR SUPER ELEMENT DECK.

COL. 1 TO 5 (SEL NO.): SUPER ELEMENT IDENTIFICATION NUMBER.

6 TO 10 (NODE NO.): ARBITRARY NUMBERING OF SUPER ELEMENT BOUNDARY NODES. MAY OR MAY NOT BE THE

```
C                                   SAME NUMBERING SCHEME AS
C                                   FOR CONNECTIVITY.
C
C          11 TO 15    (NCT):    TYPE OF COORDINATES. IF THIS FIELD IS
C                      BLANK OR ZERO, CARTESIAN COORDINATES ARE LOADED
C                      IN FORMAT DESCRIBED BELOW. NCT=(1) FOR MID SIDE
C                      NODE OF QUADRATIC SUPER ELEMENTS WITH PARTICULAR
C                      SIDE LINEAR. NCT=(2) FOR MID SIDE NODES OF CUBIC
C                      SUPER ELEMENTS WITH PARTICULAR SIDE LINEAR. LOAD
C                      ZERO OR BLANK FOR X,Y AND Z COORDINATES WHEN NCT
C                      IS (1) OR (2).
C
C          16 TO 35    X COORDINATE OF NODE, REAL*8
C
C          36 TO 55    Y COORDINATE OF NODE, REAL*8
C
C          56 TO 75    Z COORDINATE OF NODE, REAL*8
C
C      ADDITIONAL PROBLEMS MAY ALSO BE LOADED AFTER FIRST PROBLEM.
C
C      STORAGE REQUIREMENTS ARE AS FOLLOWS:
C        160K FOR NO PLOT; 180K FOR CALCOMP.
C        REPRESENTATIVE EXAMPLES USING AROUND EIGHT SUPER ELEMENTS
C        EXECUTED IN APPROXIMATELY 1 MINUTE AND 10 SECONDS.
C
C*******************************************************************
```

LIST OF REFERENCES

1. Brock, J.E., Stress Analysis of Thermowells, an internally documented report, Naval Postgraduate School, NPS-59BC74112A, Monterey, 11 Nov. 1974.

2. Adamek, J.R., An Automatic Mesh Generator Using Two and Three-Dimensional Isoparametric Finite Elements, Master's Thesis, Naval Postgraduate School, Monterey, 1973.

3. Leonidas, E., A General Purpose Three Dimensional Stress Analysis Program, Master's Thesis, Naval Postgraduate School, Monterey, 1971.

4. Lew, G.T., A Three Dimensional Solution of the Transient Field Problem Using Isoparametric Finite Elements, Masters Thesis Naval Postgraduate School, Monterey, 1972.

5. Cantin, J.G., PLISOP, an internally documented FORTRAN deck, Naval Postgraduate School, Monterey, 1972.

6. Zienkiewicz, O.C., The Finite Element Method in Engineering Science, McGraw-Hill, 1971.

7. Desai, S. and Abel, J.F., Introduction to the Finite Element Method a Numerical Method for Engineering Analysis, p. 356. Van Nostrand Reinhold, 1972.

8. Jacobsen, L.S. and Ayre, R.S., Engineering Vibrations, McGraw-Hill, 1958.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Documentation Center Cameron Station Alexandria, VA 22314	2
2.	Library, Code 0212 Naval Postgraduate School Monterey, CA 93940	2
3.	Department Chairman, Code 59Nn Department of Mechanical Engineering Naval Postgraduate School Monterey, California 93940	2
4.	Professor J.E. Brock, Code 59Bc Department of Mechanical Engineering Naval Postgraduate School Monterey, CA 93940	1
5.	LT Howard L. Crego, USN 29 Harwich Street Kingston, New York 12401	1

Thesis 160890
C8554 Crego
c.1 Fundamental vibrational frequency by a 3D - FEM procedure.